UNIVERSITY OF TWENTE

MASTER THESIS

Analysis of path-dependency in option value enhancement

Author:
Zhanxu Liu

Supervisor:
Dr. B.Roorda,
Dr. R.A.M.G. Joosten

Falculty of Behavioural, Management and Social sciences
University of Twente

November 6, 2019

UNIVERSITY OF TWENTE

Abstract

Falculty of Behavioural, Management and Social sciences
University of Twente

Master of Science

Analysis of path-dependency in option value enhancement

by Zhanxu Liu

Inspired by the market value enhancement concept of Conic hedging, we conduct an experiment to explore the contribution of path-dependency in discrete-time hedging to market value improvement of European vanilla options, in 3-step trinomial tree models. Our experiment values bid and ask prices improvement transforming hedging method from path-independent to path-dependent, under a family of exponential utility functions. To model a slightly more realistic hedging process than Conic finance, we introduce entropic risk measure to simulate the bid-ask spread of underlying in the actual market. The results show that, the improvement of bid and ask prices is not significant, under path-dependent hedging. However, we believe the model optimization in terms of optimizing algorithms and capacity of handling large data set current model is an intriguing topic worth further researching.

Keywords: Conic Finance, Dynamic hedging, path-dependency, Two-price framework, Option pricing, value enhancement, exponential utility, entropic risk measure.

Acknowledgements

Even one only teaches you one day,
you should respect one as
father/mother for lifelong.

As the Chinese old saying, I would like express my sincere gratitude to the people who taught me knowledge and gave me support during my study in the Netherlands. I would like to thank Mr.ten Naple's consistent and great support from the admission to my graduation. I would also like to thank Prof. Roorda for the patient instruction during my thesis project in the past 6 months, all the friends, who helped me during my overseas study, finally, my parents who encouraged me to take an adventure in the other side of the world. I own all of you a great debt of gratitude.

16-10-2019

Zhanxu Liu

Contents

List of Figures

List of Tables

List of Abbreviations

LOOP	Law of One Price
FTAP	Fundamental Theory of Asset Pricing
SLSQP	Sequential least squares programming
ITM	In the money
OTM	Out of the money
TTM	Time to maturity
MSE	Mean squared errors

1 Research Design

We adopted the research design proposed by Verschuren Verschuren, Doorewaard, and Mellion (2010). First, we introduce the context of our research question in Research Context (Section 1.1), Research Objective (Section 1.2) and Research Questions (Section 1.3). In the Experiment Design (Section 1.4) we propose our technical design in order to carry out our study.

1.1 Problem Context

Traditional Discrete Risk-neutral Measures of financial derivatives were built upon two fundamental assumptions:

1. The market is arbitrage-free.
2. The market is perfectly liquid.

These two assumptions present the risk-neutral valuation and under additional assumption of completeness, lead to a market where claims can be perfectly hedged and have only risk-neutral price. However, what we observe from the actual financial market is that financial instruments are traded on two prices, which are the bid(sell) price and the ask(buy) price.

The traditional pricing theory deviates from reality and more realistic theories that are built for two-price framework were later proposed. Conic Finance, is one of the new two-price finance theories proposed (Madan and Cherny, 2010). Its hedging methodology (Conic Hedging) presents a new perspective of derivative pricing. It focuses on maximizing the bid price and minimizing the ask price by hedging. Thus the market value of a financial instrument is enhanced. In the book *Applied Conic Finance* (2016), the basic conic hedging is conducted in the traditional trinomial tree model. The feature of path-dependency has not been explored, and therefore we decided to test whether path-dependent hedging would contribute in producing a significantly better market value of option, than path-independent hedging. Furthermore, to realize a more realistic result, we also introduce a new assumption, the underlying of derivative also has bid-ask spread. With these settings, the advantages of path-dependent hedging, if there are, would present in our experiment.

1.2 Thesis Structure

Our thesis consists of Five chapters:

- **Research Design** : The structure of this thesis.
- **Literature review** : Literature study about development of key theories involved in our research topic.
- **Experiment Design**: The procedure of our experiment and theoretical model introduction.
- **Experiment implementation**: Details of experiment implementation with data and quantitative analysis.
- **Conclusions and recommendation**: Conclusion of our research result and suggestions to further study in this topic,

1.3 Research Objective

This thesis aims at testing whether path-dependent hedging contributes to value enhancement of options in the actual market quotation data. Our main research objective includes three sub-objectives:

1. Design a program that conducts both path-dependent and path-independent hedging in the trinomial tree.
2. Find the optimal hedging strategy in both hedging methods by Python optimization solver.
3. Test the program with real market quotation to see whether there is any proof that path-dependent hedging significantly contributes to the value enhancement of options.

1.4 Research Questions

To reach our research objectives, we generate a main research question:
Does path dependency of hedging contribute to the option price enhancement (maximizing bid and minimizing ask) ?
This consists of multiple sub-questions that we need to answer separately.

1. Traditional discrete-time pricing model
   - What are the fundamental theorems for discrete-time asset valuation?
   - What are the assumptions of one price law?
   - What are the classical pricing models?

   These three questions give a recapitulation of classic asset pricing theories in one-price world.

2. Two-price framework
   - What are the building blocks of two-price framework?
   - What contributes to bid-ask spreads?

- What is the market value of derivatives in Conic Finance?
- What pricing models does Conic Finance use?

The answers to these sub-questions provide an overview of recent research done in the two-price world and the relation to one-price theories. The concept and formation of bid-ask spread is also defined in this section.

3. Hedging in two-price framework

- How does hedging enhance the option market value?
- What is conic hedging?
- What is the difference between conventional dynamic hedging and conic dynamic hedging?

This is the literature review about the main concept we work on, the market value enhancement of options by hedging. We explain how the price of option can be optimized by conic hedging.

4. Experimentation

- Can we reach the market value enhancement by conventional hedging?
- What is path-dependency?
- What are the assumptions in conic hedging we can continue on in our own hedging methodology?
- Which assumptions we should make to test the path dependency?
- What are the important parameters we need to tune?
- Can we find empirical evidence that shows path-dependent hedging performs statistically better than path-independent hedging?

This final section touches our central research question. we first answer the sub-questions of market value enhancement by conventional dynamic hedging, which shows path-dependency in the trinomial tree model. Then we define the key parameters and assumptions applied in our experiment. At last we test with the empirical data to see if we can find proof for the path-dependency in hedging.

1.5 Thesis outline

This is an overview of chapters in the journey to our research objective.

1. **Chapter 2: Literature review**: In this chapter, we review the related research done in our topics.
2. **Chapter 3: Experiment design**: Subsequently, we design our experiment based on the knowledge gained in last chapter.
3. **Chapter 4: Experiment implementation**: we test the empirical data of $SP500$ options in the program designed and present results.
4. **Chapter 5: Conclusion and limitation**: Based on the results presented in last chapter, we conclude some findings and also state limitation regarding our methodologies, tools, and procedures.
5. **Chapter 6: Recommendation for future research**: In the last chapter, we provide some suggestions for future study in terms of our main topic.

2
Literature review

For our literature review we adopted the process proposed by Webster and Watson (2002). First, we start with developing key theories related to our research questions and search relevant articles. Next, we go backwards by reviewing the citations used by the articles searched. Finally, we go forward by identifying articles that cite the key articles found in previous steps.

2.1 Fundamental theorems of discrete-time asset pricing

In traditional discrete-time asset pricing theories, two fundamental theorems (FTAP) have been established (Dalang, 1990; Delbaen and Schachermayer, 1994), preceding the risk-neutral measure:

- **The First Fundamental Theorem of Asset Pricing**: A discrete market is arbitrage-free and only if there exists at least one risk neutral probability measure Q on a discrete probability space ($\Omega, \mathcal{F}, P$). By definition of risk-neutral measure, the discounted price process is a martingale.
- **The Second Fundamental Theorem of Asset Pricing**: An arbitrage-free market is complete if this probability measure Q is unique. Every derivative can be replicated by the underlying securities and risk-free bond.

One of the most popular application under these two theorems is risk-neutral measure, where each underlying price is exactly equal to the discounted expectation of the underlying price. These 2 fundamental theorems are also the foundation in the traditional continuous-time asset pricing model. Models in the Black-Scholes framework also follow this setting (Merton, 1973; Black, 1973; Harrison and Kreps, 1979; Madan, 1998).

2.2 Option pricing tree model

There are multiple pricing models regarding derivative pricing, ranging from binomial to multinomial. In our thesis, we focus on tree models. Tree model is a type of traditional numerical method of asset valuation to approximate the Black-Scholes model.

2.2.1 Binomial tree

The very first tree model was formalized in 1979 to approximate the Black-Scholes option pricing (Cox, 1979; Rendleman Jr and Bartter, 1979). Binomial tracks the evolution of underlying price in lattice-based (discrete-time) manner (As Figure A.1 shown). See Appendix A.1 for the detailed parameter settings.

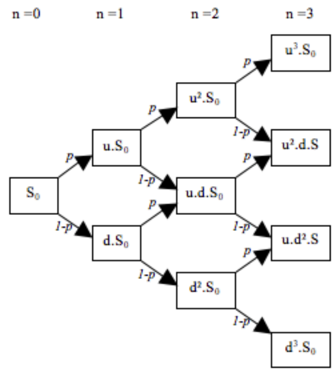

FIGURE 2.1: Binomial Tree

There are 3 steps to formalize a tree model in terms of option pricing (Cox, 1979):

- Generate underlying financial instrument price, starting from the mother node at time=0.
- Calculate the option price at each end node.
- Calculate backwardly at preceding nodes until the mother node, where the result is the price of the option

2.2.2 Trinomial tree

Multinomial cases can be developed from the basis of binomial tree, because multiple price child nodes can be created from the same mother node. In our thesis, we focus on the trinomial case of tree model, which is the trinomial option pricing tree proposed by Boyle (1986). The trinomial tree is an extension of binomial model (See Figure 2.2).

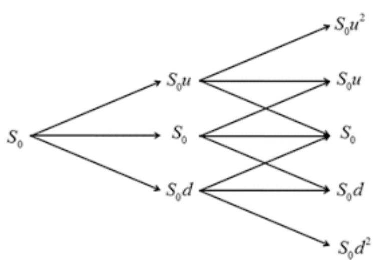

FIGURE 2.2: Trinomial Tree.

The mother node has three child nodes, which represents price moving up, remaining still and falling down respectively. Because of the state of price maintaining still, trinomial tree has the advantage of simulating the price movement of the incomplete market, which is a more realistic market[1] (Carr, 2001; Jackwerth, 1999). Binomial tree can only model complete market.

2.2.3 Tree parameters

Tree parameters usually refer to u, d and p parameter in binomial tree.R Grimwood (2000) concludes a few common parameter settings for binomial trees. In the risk-neutral model, for a small time interval ΔT, the following equations always hold:

$$S_0 e^{\Delta T} = puS_0 + (1-p)dS_0 \qquad \text{(Binomial model)}$$

$$S_0 e^{\Delta T} = p_u u S_0 + p_m m S_0 + p_d d S_0 \qquad \text{(Trinomial model)}$$

This results in different specifications of parameter selection, for instance, Cox (1979),) Jarrow and Rudd (1983), Hull and White (1988), etc. The variance of underlying price S has a variance of $S^2\sigma^2\Delta T$, which can result in a considerable variation of price level since it is positively related to large ΔT.
Furthermore, if one wants to build a tree model based on arbitrarily large steps, the underlying price S needs to be transformed into natural logarithm $x = ln(S)$, which has constant mean and variance in Geometric Brownian Motion (GBM) model. Some of the popular specifications in this setting are Trigeorgis (1991), Figlewski and Gao (1999).

2.3 Two-price framework

The law of one price (LOOP) prevails in traditional economic theories, under two hypotheses:

- **Absence of trade frictions**: Regardless of trade frictions such as transaction during asset trading.
- **Price flexibility**: In a commodity or currency market, LOOP states that identical goods (or securities) should sell for identical prices. These financial products are not in our research scope but the same statement holds in terms of option pricing.

LOOP implies that the price of a risk-free asset is the discounted expectation with respect to the so-called risk-neutral probability. A majority of classic asset pricing theories were built upon it (Ross, 1973; Chateauneuf, 1996; Harrison and Kreps, 1979; Delbaen and Schachermayer, 1994). The extensive overview of these theories is not covered in our thesis. For interested reader, see *Stochastic Finance: An Introduction In Discrete Time 2, 2004*. Despite of multiple one-price theories, we observe that single financial instrument present two prices in actual markets. These two prices are known as bid price for selling and ask price for buying. The difference of two prices is called bid-ask spread.

2.3.1 Risk preference on option pricing

The one-price assumption of traditional asset pricing models eliminates all risks in a complete market, so there is no risk over the return. In this way, all options can be

[1] A market in which Q is not unique. In such a market, a derivative can not be perfectly replicated by marketed securities but can usually be priced via the risk preferences of investors.

replicated perfectly. However, in the actual market, risks exist and all investors have risk preference during trading activities.

Risk aversion

According to Schmeidler (1989), risk aversion means that investors tend to lower the loss when they are exposed to uncertainty. To be more specific, investors would like to receive a more predictable payoff that is less than the expected payoff. For instance,the uncertain payoff a investor can receive is $X \in \{x_1, \ldots, x_n\}$ with probability $P \in \{p_1, \ldots, p_n\}$. The expected payoff of investment is $E[X] = \sum_{i=1}^{n} x_i p_i$. A risk-neutral investor is indifferent between receiving the present value $E[X]$ now or receive a uncertain outcome X in the future. If an investor is risk averse, the future payoff X is worth less than its expected value $E[X]$.

Expected-utility theory

Utility is a classic economic concept introduced by the Daniel Bernoulli in 1738, referring to the total satisfaction received from consuming a good or service. In asset pricing, it means how much the asset payoff worth based on the risk preference of investors. Continuing with the example in the last section, assuming the utility function of payoff X is $u(X)$, the utility U of payoff X is $u \cdot X \in \{u(x_1), \ldots, u(x_n)\}$. The expected utility is $E[u(X)]$ is $E[u(X)] = \sum_{i=1}^{n} u(x_i) p_i$.

Exponential utility

In our thesis, we focus on the Constant Absolute Risk Aversion (CARA) and the utility function $u(x) = 1 - e^{-\alpha x}$ family of exponential utility function represented in *Stochastic finance: an introduction in discrete time* (2011) , where α is the constant preference parameter .

2.3.2 Bid-ask spread

The bid-ask spread is the price difference between selling (bid) a financial instrument to the market and buying (ask) one from the market. It represents the liquidity of market and also the psychological difference that traders have when making decisions under risk. Theories regarding the psychological or behavioral finance side of bid-ask spread are Prospect Theory (Kahneman and Tversky, 1979) and Cumulative Prospect Theory (Tversky and Kahneman, 1992). Apart from the psychology side, there are a number of studies about the statistical estimation of bid-ask spreads (Choi, 1988; George, 1991; Roll, 1984), its component (Stoll, 1989; Huang and Stoll, 1997) and property (Gould and Galai, 1974; Klemkosky and Resnick, 1980; Kamara and Miller, 1995). All these theories assume that market acts as a passive counter party, meaning it automatically takes position opposite to the position its participants take. Based on this assumption, Madan and Cherny (2010) proposed Conic Finance. Conic finance suggests that the bid-ask spread has little connection with process, inventory, transaction costs or other formation cost, but instead reflects only the costs of holding unhedgeable risks.
Conic Finance determines the bid and ask prices of an European option based on following assumptions:

1. Not all risks can be eliminated and acceptable risks are defined the financial primitive of financial economy.

2. Zero cost cash flows at maturity of derivatives.

3. Market always acts as a passive counter party but allows prices vary with trading action (hedging).

Consider a basic example of a set of random intrinsic payoffs X to be paid out at time T, the zero-cost cash flows are:

- $X - e^{rt}b$, where traders agree to pay at Time T a cash amount $e^{rt}b$ and receive payoff. In the market's perspective, the market agrees to buy the risk X at initial cost b.
- $e^{rt}a - X$, where traders agree to receive at Time T a cash amount $e^{rt}a$ and pay out the payoff.In the market's perspective, the market agrees to sell the risk X at initial cost a.

The constant b is the bid price and the constant a is correspondingly ask price. Find out the acceptable price for b and a can be elaborated by a few steps. First, according to the traditional valuation theory, the risk-neutral value $V(X)$ of risk X at maturity T is:

$$V(X) = e^{-rt}E_Q[X] \tag{2.1}$$

Where E_Q is the risk-neutral measure and Q belongs to a convex set M. Assuming $V(X)$ is the trading price, the value Z is the cash flow of trades market find acceptable at zero cost at time T:

- At the market buying side, $Z = X - e^{rt}b$, where $b \leqq V(X)$.
- At the market selling side, $Z = e^{rt}a - X$, where $a \geqq V(X)$.

Given cash flow variable Z, we can consider a set of non-negative cash flows D, D is a convex cone under a convex risk measure probability set M:

$$D = \{Z|V(Z) = e^{-rt}E_Q[Z] \geqq 0\}, \qquad \forall Q \in M$$

To extend $V(Z)$, we have:

$$V(Z) = e^{-rt}E_Q[Z] = e^{-rt}E_Q[X - e^{rt}b] = e^{-rt}E_Q[X] - b \geqq 0 \tag{2.2}$$

$$V(Z) = e^{-rt}E_Q[Z] = e^{-rt}E_Q[e^{rt}a - X] = a - e^{-rt}E_Q[X] \geqq 0 \tag{2.3}$$

Bid and ask prices for X provided by market are given by:

$$bid(X) = e^{-rt} \inf_{Q \in M} E_Q[X] \tag{2.4}$$

$$ask(X) = e^{-rt} \sup_{Q \in M} E_Q[X] \tag{2.5}$$

Buying X equals selling $(-X)$, and thus ask price for X is the negative of bid price of $(-X)$:

$$ask(X) = -bid(-X) \tag{2.6}$$

2.4 Conic hedging

Traditional Delta hedging was first introduced in 1973 (Black, 1973; Merton, 1973). Delta hedging can replicate the payoff of an option of a underlying security, by buying or selling the security continuously before the final payoff date. In a tree model, delta hedging means trading security at each node, one step ahead of risk. In conic finance theory, the word delta hedging or so-called conic delta hedging is different from its classic delta hedging. Instead of zeroing out risk, conic hedging always seeks for a reduced convex ask price for position promised and a higher concave bid price for position held (Madan and Cherny, 2010). From the hedging strategy stand point, the ultimate purpose is to look for a Δ^{ask}_{conic} that minimizes the ask price and a Δ^{bid}_{conic} which maximizes the bid price. Under their framework, the option' risk neutral price remains the same but with a more competitive(smaller) bid-ask spread. More precisely, conic delta hedging combines both the derivative and an asset position for hedging. The optimal bid and ask price of the derivative is the price of the portfolio consisting of both derivative and cash flows from adjusting the asset position, according to Madan and Schoutens (2016).

Under the conic finance theory, one underlying owns five different prices (Madan and Schoutens, 2016). First,conic finance covers risk-neutral measure and therefore includes a risk-neutral price of an option, which is based on the Fundamental Theorem of Asset Pricing. Second, there are both unhedged ask/bid price. Without hedging activities,an option can be priced by acceptability, and by backward pricing,a unhedged ask/bid price can be generated (Madan and Schoutens, 2016). With conic hedging and backward pricing, a optimal hedged ask/bid price can be calculated in the same way as unhedged ask/bid price.

2.4.1 Market value enhancement

In continuous-time mode, a more accurate representation of general idea of conic delta hedging is to discover a portfolio V^* which at time t pays out:

$$V^*_t = f + \Delta_{conic}(S_t - e^{(r-q)dt}S_0) \tag{2.7}$$

The Δ_{conic} represents the optimal hedge strategy that yields the maximal bid and minimal ask. At $t = 0$, the value of portfolio is exactly the derivative. The market value of derivatives is considered enhanced when it receives a higher bid and a higher ask than unhedged. In a discrete-time model, for instance, binomial tree, the representation for bid price becomes:

$$V^*_{t,u} = f_{t,u} + \Delta_{conic}[u - e^{(r-q)\Delta t}]S_0 \tag{2.8}$$

$$V^*_{t,d} = f_{t,d} + \Delta_{conic}[d - e^{(r-q)\Delta t}]S_0 \tag{2.9}$$

Combing two equations,we have, under risk-neutral probability p:

$$V^*_{bid,t} = p^*_u V^*_{t,u} + p^*_d V^*_{t,d} \tag{2.10}$$

To be noticed, in Conic hedging p^*_u is the distorted value of p_u, which is computed by a distortion function $\Psi(p)$. $\Psi(p)$ generates bid-ask spread by distorting the risk-neutral probability p (Madan and Schoutens, 2016). Conic finance has proven that, in the complete market, which is represented by binomial tree, the improvement of market value is not feasible.

2.4.2 One-step Conic Delta Hedging in Two-price framework

In the conic tree models, assets are assumed to be perfectly liquid. The conic hedging also follows the same assumptions.

Conic hedging under binomial tree

In a binomial tree model, a price can only move up and down. By the fundamental theorems, this means the market is complete and the option can be replicated perfectly, where the deltas for conic ask price, conic bid price and risk-neutral price are identical. When the time step of binomial tree is infinitesimal and close to 0, the binomial tree model converges to BSM model. Thus, the absolute value of Δ^{ask}_{conic} and Δ^{bid}_{conic} converge to the Black-Scholes Δ. The delta hedging strategy is the optimal hedging strategy to reach the optimal price. According to *Applied Conic Finance*, the portfolio price for up and down states are:

$$V_u^* = f_u + \Delta_{conic}(u - e^{(r-q)\Delta T})S_0 \tag{2.11}$$

$$V_d^* = f_d + \Delta_{conic}(d - e^{(r-q)\Delta T})S_0 \tag{2.12}$$

Thus the bid price at $t = 0$ is, according to equation 2.10 :

$$V_{bid}^* = e^{-r\Delta T}[p_u^* V_u^* + p_d^* V_d^*] \tag{2.13}$$

Conic hedging under trinomial tree

Due to the existence of middle-state, where the price ends up the same during the movement, the market is incomplete in trinomial tree model. Hence the conic hedging can achieve a minimal bid and a maximum ask, whose spread is smaller than unhedged spread. Similar to binomial tree, the portfolio payoffs in three different states are:

$$V_u^* = f_u + \Delta_{conic}[u - e^{(r-q)\Delta T}]S_0 \tag{2.14}$$

$$V_m^* = f_m + \Delta_{conic}[m - e^{(r-q)\Delta T}]S_0 \tag{2.15}$$

$$V_d^* = f_d + \Delta_{conic}[d - e^{(r-q)\Delta T}]S_0 \tag{2.16}$$

Again, adjusting the equation 2.10, the bid price at time=0 is:

$$V_{bid}^* = e^{-rT}(p_u^* V_u^* + p_m^* V_m^* + p_d^* V_d^*) \tag{2.17}$$

Conic hedging under Black-Scholes model

Traditional delta hedging under the Black-Scholes model is operated in a small time step or even at in a continuous-time level, and only seeking for a risk-free position. Conic hedging shows it can achieve time value-enhanced derivative price.

2.4.3 Conic delta hedging in multi-step model

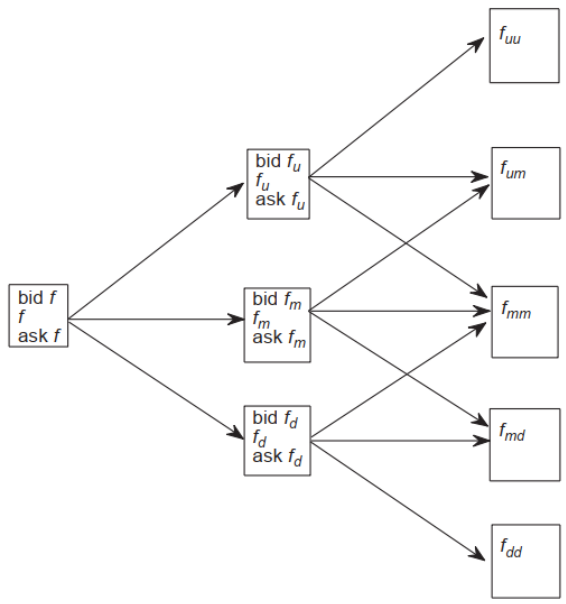

FIGURE 2.3: 2-step Recombining Trinomial Tree.

Slightly different than the one-step model, conic hedging multi-step is dynamic. For instance, in a two-step trinomial-tree model (See Figure 2.3), derivative bid price at step-1, is calculated recursively based on the intrinsic price at the end nodes:

$$bid(f_u) = e^{-rdt}(p_u^* f_{uu} + p_m^* f_{um} + p_d^* * f_{ud}) \tag{2.18}$$

$$bid(f_m) = e^{-rdt}(p_u^* f_{mu} + p_m^* f_{mm} + p_d^* * f_{md}) \tag{2.19}$$

$$bid(f_d) = e^{-rdt}(p_u^* f_{du} + p_m^* f_{dm} + p_d^* f_{dd}) \tag{2.20}$$

The final bid or ask price can computed with the same mechanism:

$$bid(f) = e^{-rdt}[(p_u^* bid(f_u) + p_m^* bid(f_m) + p_d^* bid(f_d)] \tag{2.21}$$

Then the apply and adjust the Δ at each step, a optimal ask or bid price can be obtained. For a 2-step recombining trinomial tree, there are four deltas. For an n-step recombining trinomial tree there are n^2 deltas ($\frac{3(1-3^n)}{1-3}$ for recombining trinomial tree.).

2.5 Put-Call parity

The classic put-call parity framework was first discovered in 1969, by Stoll(Stoll, 1969).The conditions of this frameworks are LOOP and the absence of market friction. In more general markets, linear pricing rules may not hold. For instance, in the real market, portfolios with the same payoffs do not always have the same formation costs. In Choquet's paper (Chateauneuf, 1996), bid-ask spread, which is one typical prevailing friction in the market, is considered. In a risk-neutral world, the put-call parity is:

$$C + Xe^{-rt} - P - S_0 e^{-qt} = 0 \tag{2.22}$$

Where X is the strike price, r is the risk-free rate, and q is the continuous dividend yield. In a two-price framework, the put-call parity is further complicated into:

$$C_{bid} + Xe^{-rt} - P_{ask} - S_0 e^{-qt} < 0 \tag{2.23}$$

Reversing the position, we have:

$$-C_{ask} - Xe^{-rt} + P_{bid} + S_0 e^{-qt} > 0 \tag{2.24}$$

Because for any derivative f, $f_{bid} \leq f \leq f_{ask}$ holds.

2.6 Conclusion on literature review

1. Multiple prevailing traditional discrete-time asset pricing theories are built to approximate price movement in the Black-Scholes-Merton framework, whose core assumptions cling to the law of one price. However, we always observe two difference bid and ask prices from one asset in the real world, where the pricing framework should be two-price.

2. Different Risk preferences of investors contribute to the creation of bid-ask spread. In the two-price framework, an investor is not risk-neutral and usually risk-averse. The degree of risk aversion is presented by the utility function of payoff.

3. Conic Finance, one of the new pricing theory in the two-price framework, stated that not all risks can be eliminated and thus define the risks that are tolerable as the financial primitive of the economy. In Conic Finance, bid and ask prices are expressed as infimum and supremum expectations under probability measures, such as risk-neutral probability.

4. Conic hedging is a recursive hedging method focusing on the market value enhancement of derivatives, instead of eliminating risks. it always looks for a set of optimal hedge parameter Δ^{conic} to maximize the bid and ask prices.

5. In the two-price framework, put-call parity does not hold like the risk-neutral world.

3
Experiment design

In this chapter, We describe the details of our experiment outline, whose models are based on Conic Finance presented in chapter 2.

3.1 Experiment outline

In our research, We follow the standard and intuitive trinomial tree model proposed by Boyle (1986) in the option pricing part and conventional dynamic hedging rules in the hedging part, instead of Conic Hedging. We keep the concept of market value enhancement and the model assumptions of conic hedging, but we choose to utilize conventional dynamic hedging, There are a few differences between our hedging method and Conic Hedging:

1. **Hedging forwardly**: Conic Hedging operates backwardly from the end nodes, which does not ensemble an intuitive hedging process and does not have path dependency. We decide to start our hedging process from the root node and therefore the hedging will be more realistic and intuitive.

2. **No probability distortion**: Conic Hedging creates the bid-ask spread by distorting the probability of price moving upwards, still and downwards. In our experiment, we simply apply exponential utility function $U(X) = 1 - e^{-\alpha x}$ and the bid-ask spread is then generated by the feature $U^{-1}(X) \neq -U^{-1}(-X)$.

3. **Introducing bid-ask spread to the underlying**: To simulate a more realistic dynamic hedging, we involve also bid-ask spread in the calculation of both option price and hedging.

3.1.1 Trinomial tree

A trinomial asset price tree will be generated by Python first (For imperfectly liquid shares, a bid price tree and a ask price tree will be generated), and then a corresponding an intrinsic option tree and a probability tree[1] (See Appendix A.4.) will be created.

[1] Each node represents its unconditional probability of price.

Having the option tree, we then input the utility function.With the utility function, the program can compute the bid and ask price under the specified utility function:

$$Price_{bid} = \sum_{i=1}^{n} u^{-1}(x_i)p_i \tag{3.1}$$

$$Price_{ask} = -\sum_{i=1}^{n} u^{-1}(-x_i)p_i \tag{3.2}$$

Where x is the final option payoff, n is the number of final price, and p is the corresponding probability on probability tree. When the utility function is linear, bid price and ask price are equal. When utility function is concave, ask price is larger than bid price.

Trinomial tree settings

Primarily, we assume underlying S is risk-neutral:

$$S_0 = E(S_T)e^{-rT} \tag{3.3}$$

Where $E(S_T)$ is valued by risk-neutral probability. Whats more, our time step ΔT is relatively small (largest ΔT=0.78 year). As a consequence, we select a popular representative of trinomial tree setting by Hull (2003).

$$u = e^{\sigma\sqrt{3\Delta T}} \tag{3.4}$$

$$d = 1/u \tag{3.5}$$

$$m = 1 \tag{3.6}$$

$$p_u = \left(\frac{e^{(r-q)(\frac{\Delta T}{2})} - e^{\sigma\sqrt{\frac{\Delta T}{2}}}}{e^{\sigma\sqrt{\frac{\Delta T}{2}}} - e^{-\sigma\sqrt{\frac{\Delta T}{2}}}}\right)^2 = \left(\frac{e^{(r-q)(\frac{\Delta T}{2})} - e^{\sigma\sqrt{\frac{\Delta T}{2}}}}{e^{\sigma\sqrt{\frac{\Delta T}{2}}} - e^{-\sigma\sqrt{\frac{\Delta T}{2}}}}\right)^2 \tag{3.7}$$

$$p_d = \left(\frac{e^{\sigma\sqrt{\frac{\Delta T}{2}}} - e^{(r-q)\sqrt{\frac{\Delta T}{2}}}}{e^{\sigma\sqrt{\frac{\Delta T}{2}}} - e^{-\sigma\sqrt{\frac{\Delta T}{2}}}}\right)^2 \tag{3.8}$$

$$p_m = 1 - p_d - p_u \tag{3.9}$$

3.1.2 Conventional dynamic hedging

The hedging calculation is different between path-dependent strategy and path-independent strategy.

Path-dependent hedging

Path-dependent hedging concerns the scenario of unrecombining trinomial tree. With the intrinsic option tree and the asset tree, the hedging activities can be conducted with a few steps. For instance, a 2-step trinomial tree hedging includes following procedure:

1. Primarily, generating a random Hedge array with 1-step tree with three hedge parameters at each node (For a n-step trinomial tree corresponds a (n-1)-step

trinomial hedge array. The numbering of variables is aligned with data. $X(i,j)$ means jth X variable at year i.

Hedge array		Asset tree			Option tree (call)		
H(0,0)	H(1,1)	S(0,0)	S(1,1)	S(2,1)	C(0,0)	C(1,1)	C(2,1)
				S(2,2)			C(2,2)
				S(2,3)			C(2,3)
	H(1,2)		S(1,2)	S(2,4)		C(1,2)	C(2,4)
				S(2,5)			C(2,5)
				S(2,6)			C(2,6)
	H(1,3)		S(1,3)	S(2,7)		C(1,3)	C(2,7)
				S(2,8)			C(2,8)
				S(2,9)			C(2,9)

TABLE 3.1: 2-step hedge.

2. After the input of hedge array, the program can calculate cash flows of hedging activities at each node:

   (a) At t=0, The cumulative cash flow is:

   $$Cashflow(0,0) = -H(0,0)S(0,0) \tag{3.10}$$

   (b) Moving from t=0 to t=1, the cumulative cash flow is:

   $$Cashflow(1,1) = -[H(1,1) - H(0,0)]S(1,1) + Cashflow(0,0)e^{rdt} \tag{3.11}$$

   For an n step model, the cumulative cash flow's calculation is similar to the probability tree, namely summing up the cumulative cash flow of mother and the hedging cash flow at current node. In formula, it is:

   $$Cashflow(n-1, node_{child}) = -[H(n-1, node_{child}) - H(n-2, node_{mother})] \\ S(n-1, node_{child}) + Cashflow(n-2, node_{mother})e^{rdt} \tag{3.12}$$

   Where the $node_{child}$ is one of the three possible outcomes from price movement of $node_{mother}$.

   (c) At maturity, the position at step $n-1$ needs to be cleared, so the cash flow at final step is:

   $$Cashflow(n, node_{child}) = H(n-1, node_{mother})S(n, node_{child}) \\ +Cashflow(n-1, node_{mother})e^{rdt} \tag{3.13}$$

3. With all cash flows computed in the hedging process, a cash flow trinomial tree is generated (See Table 3.2):

Cashflow tree		
Cash(0,0)	Cash(1,1)	Cash(2,1)
		Cash(2,2)
		Cash(2,3)
	Cash(1,2)	Cash(2,4)
		Cash(2,5)
		Cash(2,6)
	Cash(1,3)	Cash(2,7)
		Cash(2,8)
		Cash(2,9)

TABLE 3.2: 2-step cashflow tree.

The cash flow is expected to be different between the bid scenario and ask scenario, According to Equation 2.4 and 2.5. The bid price and ask price with hedging activities are, according to Equation 3.1 and 3.2 :

$$Price_{bid} = \sum_{i=1}^{n} u^{-1}(cash_i^{bid} + option_i)p_i e^{-rT} \tag{3.14}$$

$$Price_{ask} = \sum_{i=1}^{n} u^{-1}(cash_i^{ask} - option_i)p_i e^{-rT} \tag{3.15}$$

Where $option$ is the final option price (In our example call price), n is the number of end nodes, and p is the corresponding risk-neutral probability on probability tree.

Path-independent hedging

Path-independent hedging concerns the scenario of recombining trinomial tree. Due to the recombining feature of trinomial tree in this situation, the path-independent dynamic hedging does not rebalance the current position based on its mother node, if the multiple paths converge at the same node. Instead, if different paths that reaches the same price level, the cash flow of these paths is calculated by the same Hedge parameter. For instance, in the path-dependent scenario, path $up - down$, $up - down$, and $middle - middle$ result in the same price of underlying, and it has also three different hedge parameters $H_{up-down}$, $H_{up-down}$, and $H_{middle-middle}$ in the cash flow calculation. In the path-independent case, even though the calculation mechanism is the same, these 3 parameters would be equal to the same parameter $H_{middle-middle}$. Thus, the hedging of a node is independent of whichever path leads to it. In the data processing, we maintain the tree format in path-dependent hedging while setting the parameters at the same price level still.

Comparison and data format

- Path-independent hedging: Rebalancing portfolio based on underlying's current price level instead of last price level. For instance, Node 5 and Node 7 in Table 3.3 have the same underlying price. Thus the Hedge parameters are the same. In the program, these values are restricted the same.

- Path-dependent hedging: Rebalancing based on last price level (mother node). For instance Node 5 and Node 7 has different mother nodes, which are Node 1 and Node 2 respectively. Their hedge parameters are therefore different.

Timestep	0	1	2
node_map	0	1 (u)	4 (uu)
		2 (m)	5 (um)
		3 (d)	6 (ud)
			7 (mu)
			8 (mm)
			9 (md)
			10 (du)
			11 (dm)
			12 (dd)

TABLE 3.3: time step and node map of 3-step model

Table 3.3 shows the node setting of the 3-step trinomial tree. Steps 0, 1, and 2 show the three time steps in the hedging activities. This node map shows in total 13 nodes of stock states, in which the number means index and letters tracks the path of the price movement.For path-independent scenario, hedge parameters at the same price level are identical. Thus, it technically has only five hedge parameters at step 2 and nine parameters in total.

3.1.3 Introduction of bid-ask of underlying

The assumption of conic hedging is the perfect liquidity of underlyings, which means underlyings have 0 bid-ask spread. To make the hedging more realistic, we here introduced the non-perfectly liquid underlyings to the hedging. We computed the bid and ask price of the underlying by one-step ahead utility of intrinsic price level:

$$S_{t,bid} = -\frac{1}{\gamma} ln[E(e^{-\gamma s_{t+1}})] \tag{3.16}$$

$$S_{t,ask} = \frac{1}{\gamma} ln[E(e^{\gamma s_{t+1}})] \tag{3.17}$$

where in trinomial tree:

$$E(e^{-\gamma s_{t+1}}) = p_u e^{-\gamma u s_{t+1}} + p_m e^{-\gamma m s_{t+1}} + p_d e^{-\gamma d s_{t+1}} \tag{3.18}$$

With bid and ask price of shares, the final option payoffs at time T for call and put is:

$$c_T = MAX(0, S_{T,ask} - K) \tag{3.19}$$

$$p_T = MAX(0, K - S_{T,bid}) \tag{3.20}$$

3.1.4 Hedging Optimization

Optimization Tools

We select the `Scipy` package of Python to conduct our optimization. `Scipy.optimize` is one of the most popular module in black-box optimization, which is user-friendly and intuitive. One of our goals is to develop a tool to execute two hedging methods conveniently and Python is a programming language we are relatively proficient in, hence we decided to build our project around the `Scipy.optimize` of Python.

Optimization methods

For both path-dependent and path-independent scenarios, the optimization algorithm is the same. Only difference is the number of parameters in the hedge array. First, we compute the Black-Scholes delta-hedge array as initial guess as initialization. And then we utilize the Python library `Scipy.optimize.minimize` to minimize the ask price and maximize the bid price (See Appendix B.3). The `scipy`'s optimizing routine is the identical as other optimizing methods, which is iterations based on default optimizing algorithm. The optimization method in the `scipy.optimize.minimize` is Sequential Least Squares Programming (SLSQP) algorithm. SLSQP is a traditional optimization method for nonlinear problems (Kraft, 1988). It can handle the optimization with equality constraints and bounds, as well as moderately large numbers of variables(Less than 200). The detailed illustration of SLSQP is not the focus of our thesis, readers can see "A software package for sequential quadratic programming" for full details. The two optimization scenarios in the delta hedging are the maximizing bid and minimizing ask, which are mentioned in Chapter 2.

Optimization initialization

There are a few parameters that need to be set up before the optimization:

1. **Initial guess of hedge array**: To provide a good search direction and to the optimum, we first start from the traditional risk-neutral model and set an array of Black-Scholes delta, as the initial guess array for the path-independent hedging. Next, we use the result array from path-independent hedging as the initial guess of path-dependent hedging to see whether the array can be improved.

2. **Tolerance**: the minimum digits in the optimization search is 0.00001.

3. **Bounds of hedge array**: we set (-2,2) as the bound of hedge variable H, to limit the search bounds of iterations. And it turns out that the optimal hedge in some scenario can be slightly above 1 or below -1.

4. **Constraints in path-dependency**: Set equality constraints for H at the same price level in path-independent hedging. No constraints for path-dependent hedging.

3.2 Program interface

We design a one-click python program to carry out the experiment. As the Figure 3.1 shows, users can choose to input:

- S0: Initial stock price.
- Time to maturity: Number in unit of years.
- Step numbers: The step number of trinomial tree.
- Dividend yield: The optimal dividend yield of the stock

- Sigma: volatility of the option.
- Risk-free rate: libor rate.
- Position: long or short position of option.
- Option type: call, put or just asset.
- Strike: strike price
- Utility function: can be any function.

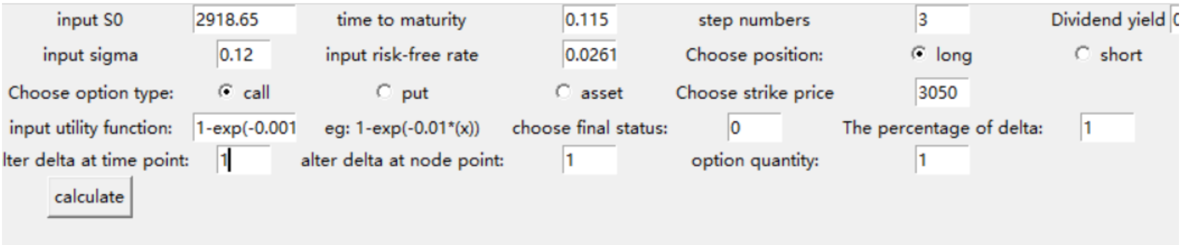

FIGURE 3.1: UI

If all parameter blanks are filled as the Figure 3.1 , the program can calculate the optimized ask and bid prices of both path-independent and path-dependent hedging and then outputs results directly. An example is shown in the code box below:

```
Spread without hedging:
[10.0733093138632 11.0616216146816]
Bid part path-independent:
10.0999578246455
[[ 0.00002 -0.45619 -0.81705]
 [ 0.      -0.079   -0.30717]
 [ 0.       0.16786 -0.0948 ]
 [ 0.       0.      -0.30717]
 [ 0.       0.      -0.0948 ]
 [ 0.       0.       0.06561]
 [ 0.       0.      -0.0948 ]
 [ 0.       0.       0.06561]
 [ 0.       0.      -1.02251]]
Ask part path-independent:
10.7540453868931
[[0.10454 0.34961 0.76242]
 [0.      0.10454 0.32348]
 [0.      0.00192 0.00198]
 [0.      0.      0.32348]
 [0.      0.      0.00198]
 [0.      0.      0.00038]
 [0.      0.      0.00198]
 [0.      0.      0.00038]
 [0.      0.      0.00039]]
Bid part path-dependent :
10.0999760375777
```

```
[[ 0.        -0.45618  -0.81705]
 [ 0.        -0.079    -0.30717]
 [ 0.         0.16785  -0.0948 ]
 [ 0.         0.       -0.30717]
 [ 0.         0.       -0.09479]
 [ 0.         0.        0.0656 ]
 [ 0.         0.       -0.09479]
 [ 0.         0.        0.0656 ]
 [ 0.         0.       -1.02251]]
Ask part path-dependent:
10.7540453243810
[[0.10454  0.34961  0.76242]
 [0.       0.10454  0.32349]
 [0.       0.00192  0.00198]
 [0.       0.       0.32348]
 [0.       0.       0.00198]
 [0.       0.       0.00038]
 [0.       0.       0.00198]
 [0.       0.       0.00038]
 [0.       0.       0.00039]]
```

As seen above, this is the output of two hedging methods and their corresponding hedge arrays. The sequence of numbers follows the ordering rules are presented by table 3.3.

4 Experiment Implementation

In this chapter, we present the detailed experiment process. We present how we process the data of market quotations and also the hedging optimization.

4.1 Data set

The options we test on are the put and call options of the S&P500 index, which is one of the most liquid securities on the global market. The option data was collected on 12th August,2019 from Yahoo Finance. The 12-month US dolloar risk free rate libor rate is 2.61%, collected from global-rates[1]. The option data includes put and call data with different time to maturities (29 days, 109 days, and 591 days separately). Only put and call options with the same strikes were collected due to our research purpose. S0 is close price of the chosen date, at 2918.65.

4.1.1 Strikes selection

Firstly, we screen out the strikes that has 0 implied volatility in only one type of option, and with only one type of option data available. We classify three strike levels, Low, ATM, and High respectively. Since the there is no strikes matching the exact spot price of underlying, we decide to select strikes in the range of $(S_0 - 100, S_0 + 100)$, more precisely, $(2818.65, 3018.65)$. For Low and High options, we choose strikes that are $\pm(200, 600)$ in intrinsic values. We prefer to choose moderately low and high strikes first, because we consider these strikes will see more substantial price enhancement than the options which are already deeply ITM (In the money) or OTM (Out of the money).

4.1.2 Parameters

There are a few parameters in the data set:

- Strike: The strike price of option.
- Bid: Bid price.

[1] https://www.global-rates.com/interest-rates/libor/american-dollar/2019.aspx

- Ask: Ask price.
- Implied Volatility: The implied volatility of put/call option on this strike.
- risk-free rate: A constant number in our experiment, which is 2.61%.

4.2 Preprocessing data

It is known that put-call parity does not hold in real markets, but the difference is still negatively related to if the liquidity of financial instruments. In this case, We adjust and find the optimal dividend that yields the minimum mean squared errors in put-call parity difference, in following steps:

1. Calculate the squared error (SE) based on Equation 2.22:

$$SE = (C + Xe^{-rt} - p - Se^{-qt})^2 \tag{4.1}$$

2. Use solver to compute the optimal q^* yielding minimum Mean Sqaured errors (MSE^*) :

$$MSE^* = \frac{1}{n}\sum_{1}^{n} SE(q^*) \tag{4.2}$$

This yields 3 $q*$:

TTM(days)	q*
29	0
109	0
591	0.0004

TABLE 4.1: Optimal dividend yield.

3. First, we select the strikes within the range and then pick one strike with lowest put-call parity squared error in each class to test.

<table>
<tr><th rowspan="2">TTM(days)</th><th rowspan="2"></th><th colspan="3">Strikes</th></tr>
<tr><th>Low</th><th>ATM</th><th>High</th></tr>
<tr><td>29</td><td></td><td>2700</td><td>2925</td><td>3050</td></tr>
<tr><td>109</td><td></td><td>2675</td><td>2950</td><td>3150</td></tr>
<tr><td>591</td><td></td><td>2675</td><td>3000</td><td>3400</td></tr>
<tr><td rowspan="3">29</td><td>Option</td><td colspan="3">Implied volatility</td></tr>
<tr><td>Call</td><td>26.25%</td><td>16.82%</td><td>12.00%</td></tr>
<tr><td>Put</td><td>22.65%</td><td>15.03%</td><td>9.72%</td></tr>
<tr><td rowspan="3">109</td><td></td><td colspan="3">Implied volatility</td></tr>
<tr><td>Call</td><td>21.52%</td><td>16.44%</td><td>13.82%</td></tr>
<tr><td>Put</td><td>19.98%</td><td>15.21%</td><td>12.60%</td></tr>
<tr><td rowspan="3">591</td><td></td><td colspan="3">Implied volatility</td></tr>
<tr><td>Call</td><td>20.52%</td><td>15.60%</td><td>13.66%</td></tr>
<tr><td>Put</td><td>18.25%</td><td>16.03%</td><td>13.33%</td></tr>
</table>

TABLE 4.2: Strikes and implied volatility in different TTM.

4.3 Attempts with long steps standard hedging

As mentioned in Chapter 2, the number of hedging parameter *Delta* grows exponentially when the step size increases. A 10-step unrecombining trinomial tree model has 88572 *H*s in total. Hedge optimization requires a enormous number of iterations, which can take an extremely long period of time to achieve any result. Therefore we only experiment in a 3-step trinomial tree model and explore in depth in this setting.

4.4 Hedging in a 3-step trinomial tree model

Moving back to the 3-step model, to test whether path dependency of delta hedging can contribute in improving the option market value, we conduct a comparison between two methods. We set up two hedging scenarios: First, we test under the assumption that the underlying is perfectly liquid. Then we introduce the bid-ask spread of the underlying.
We take time to maturities 109 days (ttm=109) as an example. With q^*=0 At K=2850 $S_0 = 2918$ $\sigma = 0.1644$, final status at the middle node.We conduct the experiment.
We test both path-independent and path-independent hedging and set of set $\alpha = 0.001$ with Utility function : $U(x) = 1 - e^{-\alpha x}$. The call option setting is:

- ttm=109 days
- q^*=0
- $S_0 = 2918.65$
- $\sigma = 0.1644$
- r=2.61%
- step=3
- K=2950

4.4.1 Hedging without bid-ask spread in the underlying

We first simulate the scenario to see if there's discrepancy between path-independent hedging and path-dependent hedging, under the assumption that the underlying is perfectly liquid and has 0 bid-ask spread.

```
###############K=2950,alpha=0.001,call ,ttm=109  days
   #####################
Call
Spread  without  hedging:
[108.252593202483  147.314093753995]
Bid  part  path-independent:
124.534859520466
[[-0.52631  -0.86253  -0.99998]
 [ 0.        -0.50744  -0.94632]
 [ 0.        -0.13248  -0.48097]
 [ 0.         0.        -0.94632]
 [ 0.         0.        -0.48097]
 [ 0.         0.        -0.0003  ]
 [ 0.         0.        -0.48097]
```

```
 [ 0.          0.          -0.0003  ]
 [ 0.          0.           0.00002]]

Bid part path-dependent :
124.534860356593
[[-0.52591  -0.86258  -0.99999]
 [ 0.        -0.50801  -0.94657]
 [ 0.        -0.13252  -0.481   ]
 [ 0.          0.        -0.94657]
 [ 0.          0.        -0.48107]
 [ 0.          0.        -0.0002  ]
 [ 0.          0.        -0.481   ]
 [ 0.          0.        -0.0002  ]
 [ 0.          0.         0.00002]]
```

Take bid price as an example, we see under the optimal hedging solution, both methods have nearly identical prices. The hedge arrays are also similar (See Appendix C for detailed hedge arrays for ask price).

Changing risk aversion parameter α

We further tune the α to 0.1,0.01, 0.05, and 0.005 to see whether there is discrepancy between paths. It can be seen in the table 4.2 (For more detailed information, please see Appendix C.1.1) that there's no significant improvement [2] of both prices under the path-dependent hedging method even with the increase of α. We also observe the spread of unhedged and the spread of hedged price are positively related to α.

Option information: Call, K=2950,TTM=109 days					
alpha	0.001	0.005	0.01	0.05	0.1
path-independent bid	124.53486	119.41685	112.17370	61.04042	45.24578
path-dependent bid	124.53486	119.41685	112.17370	61.04042	45.24578
Improvement	0.00000	0.00000	0.00000	0.00000	0.00000
path-independent ask	126.87745	131.18032	135.96383	161.85272	175.97720
path-dependent ask	126.87745	131.18032	135.96382	161.85272	175.97720
Improvement	0.00000	0.00000	0.00001	0.00000	0.00000

TABLE 4.3: Hedge results in different α

4.5 Introducing bid-ask spread in shares

As mentioned in last chapter, bid-ask spread in shares is involved in our option pricing and hedging. During the hedging, the program buys the underlying at ask price and sells it at bid price. The cash flow calculation is therefore,before last step:

- If buying:

$$cash_{child} = (\Delta_{child} - \Delta_{mother})S_{child,ask} + cash_{mother}e^{rdt}, \ (\Delta_{child} - \Delta_{mother}) \geq 0 \tag{4.3}$$

[2]We define the price difference between path-dependent and path-independent hedging as the improvement. For bid price, it's $[bid(path - dependent) - bid(path - independent)]$. For ask price, it's $-[ask(path - dependent) - ask(path - independent)]$.

- If selling:

$$cash_{child} = -(\Delta_{child} - \Delta_{mother})S_{child,bid} + cash_{mother}e^{rdt}, \ (\Delta_{child} - \Delta_{mother}) < 0 \tag{4.4}$$

At last step, to clear the position:

- If buying:

$$cash_{child} = \Delta_{mother}S_{child,ask} + cash_{mother}e^{rdt}, (\Delta_{mother}) \geq 0 \tag{4.5}$$

- If selling:

$$cash_{child} = \Delta_{mother}S_{child,bid} + cash_{mother}e^{rdt}, (\Delta_{mother}) < 0 \tag{4.6}$$

4.5.1 Effect of parameter γ in entropic risk measure

For $S_0 = 2918.65$, $\gamma > 0$, the smaller the γ is, the narrower the spread in shares gets:

- For $\gamma = 0.0001$, the spread is [2926.336,2929.781].
- For $\gamma = 0.001$, the spread is [2910.974,2945.421].

The optimization soon breaks in a few searches if the spread is large, like the example in $\gamma = 0.001$. The reason is that the cost of trading the underlying is huge. With this in mind, we choose $\gamma = 0.0001$ with a small spread of bid-ask in the underlying to ensure the hedging program can operate. In this setting, the bid price and ask price are both slightly higher than the intrinsic price.

4.5.2 Hedging results

In this section, we utilize the bid-ask equations for the underlying and carry out the optimization. The parameter γ is set to 0.0001 and the parameter α is set to 0.001. The table below shows the hedge result of $U(x, \alpha)$.

TTM:29 days			
Strikes	2700	2925	3050
Call_bid Improvement	0.00006	0.00002	0.00002
Call_ask Improvement	0.00000	0.00000	0.00001
Put_bid improvement	0.00000	0.00000	0.00000
Put_ask improvement	0.00000	0.00000	0.00001
TTM:109 days			
Strikes	2675	2950	3150
Call_bid Improvement	0.00005	0.05606	0.00004
Call_ask Improvement	0.01880	0.00000	0.00001
Put_bid improvement	0.00941	0.00500	0.00000
Put_ask improvement	0.00000	0.00012	0.00005
TTM:591 days			
Strikes	2675	3000	3400
Call_bid Improvement	0.00006	0.00032	0.00267
Call_ask Improvement	0.00042	0.00000	0.06642
Put_bid improvement	0.00001	0.03321	0.03186
Put_ask improvement	0.00034	0.00002	0.00028

TABLE 4.4: Hedge results, α=0.001.

The discrepancy between two methods is not prominent but we see longer TTM and higher strike seem to yield relatively large improvement. In addition to that, the improvement of bid and ask prices appears to be more prominent than the scenario with perfectly liquid underlying in Table 4.3.
We choose the strike (K=3400) with most prominent discrepancy to test whether discrepancy will increase with different α (See Appendix C.2.1 for full information).

α =0.01, Strike=3400,ttm=591 days		
	Call	Put
path-independent bid	70.23979955	334.250769
path-dependent bid	70.24889797	334.2528231
Improvement	0.00910	0.00205
path-independent ask	208.2734201	614.5963534
path-dependent ask	208.2675762	614.5952339
Improvement	0.00584	0.00112

TABLE 4.5: Tuning α to 0.01.

We see the spread after hedging is already huge after the increase of α, but the enhancement effect of path-dependency is still minor.

α =0.0001, Strike=3400		
	Call	Put
path-independent bid	132.5901558	405.0719549
path-dependent bid	132.5901558	405.0719557
Improvement	0.00000	0.00000
path-independent ask	133.4295909	405.8928119
path-dependent ask	133.4295912	405.8928119
Improvement	0.00000	0.00000

TABLE 4.6: Tuning α to 0.0001.

If we tune down the α, the improvement is almost zero. When the unhedged spread is small, the improvement that path-dependent hedging can achieve becomes small.

4.5.3 Main Findings

1. Increasing α (As well as increasing unhedged spread.) does not necessarily yield a more prominent price enhancement in scenario with and without perfectly liquid underlying.

2. In the non-perfectly liquid underlying scenario, options with longer time to maturity seem to have more prominent path-dependent effect than options with extremely short time to maturity (29 days).

3. There seems to be a hidden non-linear optimal combination of α and strikes because the path-dependent effect is not linear with the strike. This might be worth further investigation,which can include even with more parameters like γ, u, d, m and p etc.

4. In 3-step trinomial tree, the path-independent hedging can optimize itself to get close to the optimum, which should be reached by path-dependent hedging. This could be due to:

   - **Limited number of constraints**: In our 3-step model, there are only four equality constraints. 4 constraints might have limited impact on the final result. We can expect more if we increase the steps.
   - **Absence of penalty**: There is no penalty for path-independent hedging. increasing the weights of recombining end nodes in the utility function can see more discrepancy. A change of option type, for instance, to butterfly, might see more difference.

5 Conclusion and limitation

This chapter presents our conclusion of experiment and limitations we face during the the research.

5.1 Conclusion

Through our experiment, we only found minor effects of market value enhancement path-dependency in our setting. However, we did observe increases in improvement with the time to maturity increases.

5.2 Limitations

The market value enhancement concept of Conic Hedging is a relatively new and has yet to be fully explored. We give spread to the underlying around this concept and try to see a market value enhancement that is closer to the actual market. Since there is few studies with the same setting, we can say that our research is an adventure in this topic. There are a few limitations that hinder us from getting more realistic results:

- **Computation power for large data set**: As we mentioned in Section 4.3, the number of nodes in the trinomial tree grows exponentially, and so does the time of optimization. Conducting the optimization in clouds or with high-performance processors might give optimum result.
- **Optimizer**: The `SLSQP` methods can handle variables with a limited number and meanwhile majority of optimization methods in `Scipy` cannot handle optimization with equality/inequality constraints and bounds. Expanding the research to a higher level Python library will also lead to better results.
- **Objective function**: We keep our gradient function to be optimize as clear as it can be, but our knowledge in programming might cause us creating an objective function that hides multiple plateaus. The current program stops iterating while hitting plateaus multiple times and this can prevent us from finding the true global optimum.

- **Optimal α and γ**: We observe that the improvement of option market value also relies on the levels of parameter α and γ. Finding the suitable α and γ will see a more prominent discrepancy.

- **Other hedge parameters**: Better Jacobian matrix, gradient function, and initial guess can lead to more precise and accurate results. Customization of these parameters consumes considerable time and effort for someone without mathematical optimization background.

- **Other Utility functions**: Although exponential utility function is a popular function to model risk aversion, it only presents one attitude towards risk. In the dynamic hedging process, market's risk aversion could vary. Implementation of a dynamic utility function which adjust its risk parameter at different time steps or price levels can realize a more realistic process.

6
Recommendation of future study

During our research, many new ideas came up and we list a few of them that might worth exploring:

1. **Utility Function**: The applied exponential utility function is related to constant relative risk aversion, but other utility function regarding relative absolute risk aversion can also be intriguing.

2. **Longer steps**: As mentioned in the last chapter, Longer steps will generate much more discrepancy between 2 methods, based on the effect of cash flows accumulation.

3. **Combination of path-independent hedging and path-dependent hedging**: Our assumption is program either execute hedging in a full path-dependent way or full path-independent way. The scenarios of a mixture of these 2 methods is also worth research and we believe this also test the sensitivity of hedging result on difference paths in trinomial tree.

4. **Diversity of options**: Besides vanilla European options, other types options might have variations in bid-ask spread when path-dependency of hedging changes.

5. **Tree parameter setting**: Our tree setting is suitable for only for small time interval, which can generate some variance for options with long contract period. We believe a new specification which has constant variance is beneficial.

6. **Suitable bid-ask model for underlying**: Other sequentially consistent models apart from entropic risk measure, for bid and ask prices computation of underlying.

7. **Multinomial tree model**: We conjecture that there will be discrepancy between two methods if we increase the child nodes in the tree model, because the number of paths also increases. But this also requires a more efficient program and stronger computation power.

A
Models and algorithms

A.1 Appendix: Binomial tree setting

$$u = e^{\sigma\sqrt{\Delta T}} \tag{A.1}$$

$$d = 1/u \tag{A.2}$$

$$p_u = \frac{e^{rt/n} - d}{u - d} \tag{A.3}$$

$$p_d = 1 - p_u \tag{A.4}$$

$$S_u = S_0 u \tag{A.5}$$

$$S_d = S_0 d \tag{A.6}$$

A.2 Appendix: Popular trinomial tree setting

$$u = e^{\sigma\sqrt{3\Delta T}} \tag{A.7}$$

$$d = 1/u \tag{A.8}$$

$$m = 1 \tag{A.9}$$

$$p_u = \left(\frac{e^{(r-q)(\frac{\Delta T}{2})} - e^{\sigma\sqrt{\frac{\Delta T}{2}}}}{e^{\sigma\sqrt{\frac{\Delta T}{2}}} - e^{-\sigma\sqrt{\frac{\Delta T}{2}}}} \right)^2 \tag{A.10}$$

$$p_d = \left(\frac{e^{\sigma\sqrt{\frac{\Delta T}{2}}} - e^{(r-q)\sqrt{\frac{\Delta T}{2}}}}{e^{\sigma\sqrt{\frac{\Delta T}{2}}} - e^{-\sigma\sqrt{\frac{\Delta T}{2}}}} \right)^2 \tag{A.11}$$

$$p_m = 1 - p_d - p_u \tag{A.12}$$

$$S_u = S_0 u \tag{A.13}$$

$$S_d = S_0 d \tag{A.14}$$

$$S_m = S_0 m \tag{A.15}$$

A.3 Appendix: Conic trinomial tree setting

$$u = e^{[(r-\sigma^2/2)dt+\sigma\sqrt{3dt}]} \tag{A.16}$$

$$d = 1/u \tag{A.17}$$

$$m = e^{[(r-\sigma^2/2)dt]} \tag{A.18}$$

$$p_u = 1/6 \tag{A.19}$$

$$p_d = 1/6 \tag{A.20}$$

$$p_m = 1 - p_d - p_u \tag{A.21}$$

$$S_u = S_0 u \tag{A.22}$$

$$S_d = S_0 d \tag{A.23}$$

$$S_m = S_0 m \tag{A.24}$$

A.4 Appendix: Probability tree setting

A probability tree is a trinomial tree with unconditional probability of option price at each node. A probability of a child node is the production of probability that at previous nodes that lead to this child node. An example is shown below:

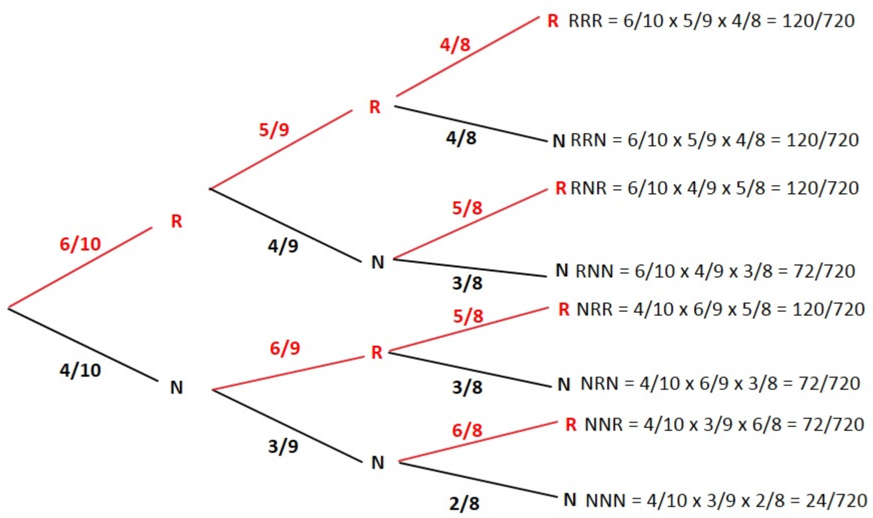

FIGURE A.1: Probability Tree

A.5 Appendix: Sequential least squares programming

SLSQP algorithm is an iterative method for solving constrained nonlinear optimization problems (Meanwhile supporting both inequality and equality constraints). For our research we have a set of hedge variables $[x_1, ..., x_{13}]$, which are constrained by following conditions:

- inequality constraints: $-1 <= X <= 1$

- For the path-independent hedging, we have equality constraints for hedge parameter x at the same price level: $x_i = x_j, if\,price_i = price_j$

The algorithm solves a optimization in the form of:

$$\min_x = f(x) \tag{A.25}$$

Where x in our case is a sequence of array. It also subject to inequality and equality constraints like:

$$b(x) \leq c \tag{A.26}$$

$$b(x) = c \tag{A.27}$$

$$a \leq x \leq b \tag{A.28}$$

Where c is a constant number. These constraints can also be set to the elements in x. The size of the problem should be only moderately large, in our case, Length of x should be less than 200 (Kraft, 1988). The basic algorithm is

$$x^{k+1} = x^k + \alpha^k d^k \tag{A.29}$$

Where k is the number of step, d^k is the search direction,α^k is the step length. The search direction is in the form of Lagrangian function:

$$L(x, \lambda) = f(x) - \sum_{j=1}^{n} \lambda_j g_j(x) \tag{A.30}$$

Which is a mathematical function to find the local maxima and minima.

B
Python code

B.1 Appendix: trinomial tree generating

```
import numpy as np
import pandas as pd
import sympy as sy
import scipy.optimize as optimize
from pulp import *

#Genrating option class to store parameters
class option:
    def __init__(self, T, steps, vol, s0, r, optiontype,
       strike, position,div,qty):
        self.ttm = T
        self.steps = steps
        self.vol = vol
        self.r = r
        self.s=s0
        self.initial_price = s0
        self.optiontype = optiontype
        self.strike = strike
        self.position = position
        self.dt = T / steps
        self.div = div
        self.purer = self.r - self.div
        self.discount = np.exp(-self.purer * self.dt)
        self.rfdiscount = np.exp(-self.r * self.dt)
        self.divdiscount=np.exp(-self.div*self.dt)
        self.u = np.exp(vol*np.sqrt(2*self.dt))#np.exp((self.
           r-(vol**2)/2)*self.dt+
                        #vol*np.sqrt(3*self.dt))
        self.d = 1 / self.u
        self.m=1
```

```
        self.pu = (np.exp((self.r-self.div)*self.dt/2)-np.exp
           (-self.vol*np.sqrt(self.dt/2)))**2\
                /(np.exp(self.vol*np.sqrt(self.dt/2))-np.
                   exp(-self.vol*np.sqrt(self.dt/2)))**2
        self.pd = (np.exp(self.vol*np.sqrt(self.dt/2))-np.exp
           ((self.r-self.div)*self.dt/2))**2\
                /(np.exp(self.vol*np.sqrt(self.dt/2))-np.
                   exp(-self.vol*np.sqrt(self.dt/2)))**2
        self.pm=1-self.pu-self.pd
        self.qty=qty

#Generating trinomial tree
def trinomial_tree(option_info, optiontype, strike):
    "optiontype:+1:call -1:put" \
    "r:interest rate" \
    "s0:initial price" \
        # basic calculation
    pu = option_info.pu
    pd = option_info.pd
    pm = 1 - pu - pd
    # generate tree
    initial_tree = np.zeros([3 **(option_info.steps),
       option_info.steps + 1])
    initial_tree[0, 0] = option_info.initial_price
    for col in range(1, option_info.steps + 1):  # 1 to 2, if
        2 steps
        index = 3 **(col-1)
        for i in range(index):
                initial_tree[i*3, col] = initial_tree[i, col
                   - 1]*option_info.u*option_info.divdiscount
                initial_tree[i*3+ 1, col] = initial_tree[i,
                   col - 1]*option_info.divdiscount*
                   option_info.m
                initial_tree[i*3 + 2, col] = initial_tree[i,
                   col - 1]*option_info.d*option_info.
                   divdiscount
    stock_tree = initial_tree
    'calculate v(f)'  #checked#

    # obtion value
    if optiontype != 0:
        option_tree = np.zeros([3 **(option_info.steps),
           option_info.steps + 1])
        endnotes=3**option_info.steps
        for j in range(endnotes):
            option_tree[j, option_info.steps] = max(0,
               optiontype * (stock_tree[j, option_info.steps]
                - strike))
        for i in range(option_info.steps - 1, -1, -1):
            for j in range(3**i):
```

```
                    option_tree[j, i] = (pu * option_tree[j*3, i
                        +1] + pm* option_tree[j*3+ 1, i + 1]+
                                            pd*option_tree[j*3+2, i
                                                +1])*option_info.
                                                rfdiscount
        return option_tree
    else:
        return (stock_tree)

    # generate a matrix

def probability(option_info):
    initial_tree = np.zeros([3**option_info.steps,option_info
        .steps+1])
    initial_tree[0,0] = 1
    probability_m = 1 - (option_info.pu+option_info.pd)
    for col in range(1, option_info.steps + 1): # 1 to 2, if
        2 steps
        index = 3 **(col-1)
        for i in range(index):
                    initial_tree[i*3, col] = initial_tree[i, col
                        - 1]*option_info.pu
                    initial_tree[i*3+ 1, col] = initial_tree[i,
                        col - 1]*probability_m
                    initial_tree[i*3 + 2, col] = initial_tree[i,
                        col - 1]*option_info.pd

    return initial_tree

#input utility function
def utility(formula, **kwargs):
    "a␣is␣the␣degree␣of␣loss␣aversion,␣a>0,␣rise␣averse"
    expr = sy.sympify(formula)
    return expr.evalf(subs=kwargs)
```

B.2 Appendix: Delta Hedging

```
##############Continuing with last section################
#flatten the trinomial tree into a 1-D array
def flatten_tree(anytree):#good
    flatten_tree=np.empty([0,1])
    ttm=len(anytree[0])
    for i in range(ttm):
        #0 for 0, 1,2,3 for 0, 4-13 for 1,2,3
        nodes=3**i
        trimmed_array=anytree[:nodes,i]
        flatten_tree=np.append(flatten_tree,trimmed_array)
    return flatten_tree
```

```
###########Rebalancing mechanicsm######################
def mother_child_hedging(option_info,cashflow,last_start_index,start_index,strategy,asset_tree):
    for step in range(1,option_info.steps):#years
        temp_start_index = start_index#save current year start index into a variable
        #test=[i for i in range(last_start_index,start_index)]
        for i in range(last_start_index,start_index):#mother nodes
            for j in range(start_index,start_index+3):#3 childnodes of single mother node
                cashflow[j]=-(strategy[j]-strategy[i])*asset_tree[j]+cashflow[i]/option_info.rfdiscount #cumulative cash flow
            start_index+=3 #3 child nodes for next mother node
        last_start_index=temp_start_index# start index of year step becomes start index of step+1
    #last year
    temp_start_index=start_index
    for i in range(last_start_index,start_index): #year N-1
        for j in range(start_index, start_index + 3):
            cashflow[j] = (strategy[i]) * asset_tree[j] + cashflow[i]/option_info.rfdiscount
        start_index += 3  # 3 child nodes for next mother node

    return cashflow

###########Hedging without bid-ask in shares ##################
def hedging(option_info, asset_tree,strategy):
    cashflow = flatten_tree(asset_tree)
    asset_tree = flatten_tree(asset_tree)
    # all 1D array
    last_start_index=0
    start_index=1
    cashflow[last_start_index]=-asset_tree[last_start_index]*strategy[last_start_index]
    test=mother_child_hedging(option_info, cashflow, last_start_index, start_index, strategy, asset_tree)
    return test
```

B.3 Appendix: Maximization/minimization of bid/ask

```
##############Continuing with last section###############
from scipy import optimize
```

```
#############Create bid-ask spread by entropic risk measure
    #########
def option_bid_ask(asset_tree ,gamma,option_info):#return
    optionpayoff with
     intrinsic_payoff=asset_tree[:, -1].copy()
     for i in range(len(intrinsic_payoff)):
         intrinsic_payoff[i] = eontropic(option_info,
             intrinsic_payoff[i], gamma, -option_info.
             optiontype)#-1 for call
         intrinsic_payoff[i]=abs(intrinsic_payoff[i])#take
             positive value for ask price
         intrinsic_payoff[i]=max(0,(intrinsic_payoff[i]-
             option_info.strike)*option_info.optiontype)
     realpayoff=intrinsic_payoff
     return realpayoff
#####################expected utility##########
def recursive_hedge(hedge_result, formula,option_info,
    probability_tree):
     U_hedge=np.array(hedge_result)
     for i in range(len(hedge_result)):
         U_hedge[i]= utility(x=hedge_result[i],formula=formula)
     PV_hedge=U_hedge*(option_info.rfdiscount**option_info.
        steps)
     EU=np.dot(PV_hedge,probability_tree[:,-1])
     return EU
##################Initial Guess, create a black-scholes delta
     array#############
def Guess_delta(asset, option_info):
     asset=flatten_tree(asset)
     start = 0
     end = 1
     reblance = asset.copy()
     reblance.fill(0)
     reblance[0] = bs_delta(option_info, 0, 1, asset[0]) * -
        option_info.position
     for year in range(0, option_info.steps):
             nextyear=year+1
             temp = end  # save end mother index
             movingend = end
             for mother in range(start, end):
                 childstart = mother * 3 + 1  # childindex
                 for j in range(childstart, childstart + 3):
                    # pass the cashflow this year to next year
                     if (nextyear < option_info.steps):
                         delta_new = bs_delta(option_info,
                             nextyear, 1,
                                                  asset[j]) * -
                                                     option_info.
                                                     position  #
                                                     compute new
                                                     delta
```

```
                        reblance[j] =delta_new   # rebalance
                    else:
                        unwind_delta=bs_delta(option_info ,
                            year, 1, asset[mother])*
                            option_info.position
                        reblance[j]=unwind_delta
                movingend += 3
            end = movingend
            start = temp
    return reblance
#########Initialization###########
tol=1e-5
initial_guess=Guess_delta(asset_tree, option_info)
initial_guess=initial_guess[:13]

################ hedging without bid-ask spread in shares
    ###############

    strategy_tree=flatten_tree(asset_tree[:,:-1])#create a
        strayegy tree, #remove last column of asset tree and
        copy

    def nethedge(strategy_tree):
        option_off = option_tree[:, -1]*position
        hedge_result=p.hedging(option_info, asset_tree,
            strategy_tree)
        final_result=hedge_result[13:]+option_off#
            optionpayoff plus net hedge
        price=-p.recursive_hedge(final_result,
            utility_function,option_info,probability_tree)
        return price

bns=[(-2,2)]*len(strategy_tree)#bounds for H

# to find the maximized bid or minimized ask
list = optimize.minimize(nethedge,initial_guess,bounds=bns,
    tol=tol,,method='SLSQP')
hedgeH=np.asarray(list.x)
optimal_result=nethedge(reverse_hedge)
################ hedging with bid-ask spread in shares
    ###############
    def nethedge_spread(strategy_tree):
        option_off=p.option_bid_ask(asset_tree,gamma
                                   , option_info)*position
        hedge_result = p.hedging_cash_entropic(option_info,
            asset_tree, strategy_tree,bid_tree,ask_tree)
        # hedge_result=p.hedging(option_info, asset_tree,
            strategy_tree)
        final_result=hedge_result[13:]+option_off#
            optionpayoff plus net hedge
```

```
        price=-p.recursive_hedge(final_result,
            utility_function,option_info,probability_tree)
        return price
bns=[(-2,2)]*len(strategy_tree)#bounds for H

# to find the maximized bid or minimized ask
list_spread = optimize.minimize(nethedge_spread,initial_guess
    ,bounds=bns,,tol=tol,,method='SLSQP')
hedgeH=np.asarray(list.x)
optimal_result=nethedge(reverse_hedge)
```

C
Results

C.1 Appendix: Hedging without bid-ask spread in the underlying

Under $U(X) = 1 - e^{-\alpha * x}, \alpha = 0.001, 0.005, 0.01, 0.05$, we have:

```
################K=2950,alpha=0.001 and 0.005,call ,ttm=109
    days######################
alpha=0.001        alpha=0.005
Spread without hedging: Spread without hedging:
[108.252593202483 147.314093753995]       [65.4065691238308
    281.297137006276]
Bid part path-independent:        Bid part path-independent:
124.5348595       119.416845
[[-0.52631  -0.86253  -0.99998]   [[-0.52763  -0.8735    -0.99998]
 [ 0.       -0.50744  -0.94632]    [ 0.       -0.51048  -0.94689]
 [ 0.       -0.13248  -0.48097]    [ 0.       -0.11885  -0.48644]
 [ 0.        0.       -0.94632]    [ 0.        0.       -0.94689]
 [ 0.        0.       -0.48097]    [ 0.        0.       -0.48644]
 [ 0.        0.       -0.0003  ]   [ 0.        0.       -0.00014]
 [ 0.        0.       -0.48097]    [ 0.        0.       -0.48644]
 [ 0.        0.       -0.0003  ]   [ 0.        0.       -0.00014]
 [ 0.        0.        0.00002]]   [ 0.        0.
    -0.00006]]
Ask part path-independent:        Ask part path-independent:
126.8774532       131.1803241
[[ 0.5257    0.85689   0.99984]    [[ 0.52528   0.84602   0.99997]
 [ 0.        0.50676   0.94674]    [ 0.        0.50513   0.94699]
 [ 0.        0.13942   0.47896]    [ 0.        0.15289   0.47554]
 [ 0.        0.        0.94674]    [ 0.        0.        0.94699]
 [ 0.        0.        0.47896]    [ 0.        0.        0.47554]
 [ 0.        0.       -0.00002]    [ 0.        0.       -0.00008]
 [ 0.        0.        0.47896]    [ 0.        0.        0.47554]
 [ 0.        0.       -0.00002]    [ 0.        0.       -0.00008]
```

```
 [ 0.          0.         -0.00024]]    [ 0.          0.
     -0.00005]]
Bid part path-dependent :          Bid part path-dependent :
124.5348604        119.4168456
[[-0.52591  -0.86258  -0.99999]    [[-0.52756  -0.87354  -0.99998]
 [ 0.        -0.50801  -0.94657]     [ 0.        -0.51043  -0.94689]
 [ 0.        -0.13252  -0.481   ]    [ 0.        -0.11886  -0.48644]
 [ 0.         0.        -0.94657]    [ 0.         0.        -0.94689]
 [ 0.         0.        -0.48107]    [ 0.         0.        -0.48644]
 [ 0.         0.        -0.0002  ]   [ 0.         0.        -0.00012]
 [ 0.         0.        -0.481   ]   [ 0.         0.        -0.48644]
 [ 0.         0.        -0.0002  ]   [ 0.         0.        -0.00012]
 [ 0.         0.         0.00002]]   [ 0.         0.
     -0.00005]]
Ask part path-dependent:           Ask part path-dependent:
126.8774532        131.1803198
[[ 0.5257    0.85689   0.99984]    [[ 0.52493   0.84586   0.99998]
 [ 0.         0.50676   0.94674]    [ 0.         0.50522   0.94692]
 [ 0.         0.13942   0.47896]    [ 0.         0.15293   0.47554]
 [ 0.         0.         0.94674]   [ 0.         0.         0.94692]
 [ 0.         0.         0.47896]   [ 0.         0.         0.47555]
 [ 0.         0.        -0.00002]   [ 0.         0.        -0.00007]
 [ 0.         0.         0.47896]   [ 0.         0.         0.47554]
 [ 0.         0.        -0.00002]   [ 0.         0.        -0.00007]
 [ 0.         0.        -0.00024]]  [ 0.         0.
     -0.00004]]

###############K=2950,alpha=0.01 and 0.05,call,ttm=109 days
    #####################
alpha=0.01        alpha=0.05
Spread without hedging: Spread without hedging:
[41.1037921760627 467.850927161967]      [8.92566948742926
    763.644507653378]
Bid part path-independent:         Bid part path-independent:
112.1736989        61.04041862
[[-0.53046  -0.88619  -1.00009]    [[-0.67172  -0.93089  -0.99995]
 [ 0.        -0.51527  -0.94717]     [ 0.        -0.68498  -0.94809]
 [ 0.        -0.10321  -0.49594]     [ 0.        -0.04621  -0.6976  ]
 [ 0.         0.        -0.94717]    [ 0.         0.        -0.94809]
 [ 0.         0.        -0.49594]    [ 0.         0.        -0.6976  ]
 [ 0.         0.        -0.00017]    [ 0.         0.        -0.00142]
 [ 0.         0.        -0.49594]    [ 0.         0.        -0.6976  ]
 [ 0.         0.        -0.00017]    [ 0.         0.        -0.00142]
 [ 0.         0.         0.00025]]   [ 0.         0.
     -0.00008]]
Ask part path-independent:         Ask part path-independent:
135.9638305        161.8527197
[[ 0.52466   0.8325    1.0001  ]    [[ 0.52619   0.78118   1.00001]
 [ 0.         0.50408   0.94666]    [ 0.         0.50237   0.94603]
 [ 0.         0.16865   0.47259]    [ 0.         0.22474   0.46453]
 [ 0.         0.         0.94666]   [ 0.         0.         0.94603]
```

```
 [ 0.          0.          0.47259]
 [ 0.          0.         -0.00018]
 [ 0.          0.          0.47259]
 [ 0.          0.         -0.00018]
 [ 0.          0.          0.00024]]
Bid part path-dependent :
112.1737004        61.04041879
[[-0.53035  -0.88623  -1.00007]
 [ 0.        -0.51528  -0.94711]
 [ 0.        -0.10311  -0.49594]
 [ 0.          0.        -0.94711]
 [ 0.          0.        -0.49591]
 [ 0.          0.        -0.00012]
 [ 0.          0.        -0.49594]
 [ 0.          0.        -0.00012]
 [ 0.          0.          0.00023]]
Ask part path-dependent:
135.9638234        161.8527186
[[ 0.52484   0.83255   1.00009]
 [ 0.          0.50401   0.94665]
 [ 0.          0.16862   0.47257]
 [ 0.          0.          0.94665]
 [ 0.          0.          0.47256]
 [ 0.          0.         -0.00016]
 [ 0.          0.          0.47257]
 [ 0.          0.         -0.00016]
 [ 0.          0.          0.00023]]
```

```
 [ 0.          0.          0.46453]
 [ 0.          0.         -0.00001]
 [ 0.          0.          0.46453]
 [ 0.          0.         -0.00001]
 [ 0.          0.
   -0.00001]]
Bid part path-dependent :
[[-0.67168  -0.93089  -0.99996]
 [ 0.        -0.68496  -0.94811]
 [ 0.        -0.04621  -0.6976 ]
 [ 0.          0.        -0.94812]
 [ 0.          0.        -0.69761]
 [ 0.          0.        -0.00141]
 [ 0.          0.        -0.6976 ]
 [ 0.          0.        -0.00141]
 [ 0.          0.
   -0.00008]]
Ask part path-dependent:
[[ 0.52622   0.78119   1.00001]
 [ 0.          0.50237   0.94603]
 [ 0.          0.22474   0.46453]
 [ 0.          0.          0.94603]
 [ 0.          0.          0.46453]
 [ 0.          0.         -0.00001]
 [ 0.          0.          0.46453]
 [ 0.          0.         -0.00001]
 [ 0.          0.
   -0.00001]]
```

```
 ################K=2950,alpha=0.1,call#####################
alpha=0.1
Spread without hedging:
[4.46286065256616  803.558896262558]
Bid part path-independent:
45.24577531
[[-0.79232  -0.94167  -1.       ]
 [ 0.        -0.79239  -0.9505  ]
 [ 0.        -0.10033  -0.79049]
 [ 0.          0.        -0.9505  ]
 [ 0.          0.        -0.79049]
 [ 0.          0.        -0.03127]
 [ 0.          0.        -0.79049]
 [ 0.          0.        -0.03127]
 [ 0.          0.        -0.00017]]
Ask part path-independent:
175.9772018
[[ 0.52563   0.76772   0.99997]
 [ 0.          0.50396   0.9454  ]
 [ 0.          0.23771   0.46291]
```

```
 [ 0.         0.         0.9454  ]
 [ 0.         0.         0.46291]
 [ 0.         0.        -0.00012]
 [ 0.         0.         0.46291]
 [ 0.         0.        -0.00012]
 [ 0.         0.        -0.00001]]
Bid part path-dependent :
45.24577533
[[-0.79232  -0.94167  -1.       ]
 [ 0.       -0.79238  -0.95049]
 [ 0.       -0.10033  -0.79049]
 [ 0.        0.       -0.95049]
 [ 0.        0.       -0.7905  ]
 [ 0.        0.       -0.03127]
 [ 0.        0.       -0.79049]
 [ 0.        0.       -0.03127]
 [ 0.        0.       -0.00017]]
Ask part path-dependent:
175.9772
[[ 0.52561   0.76771   0.99998]
 [ 0.        0.50396   0.9454  ]
 [ 0.        0.23773   0.46291]
 [ 0.        0.        0.9454  ]
 [ 0.        0.        0.46291]
 [ 0.        0.       -0.00012]
 [ 0.        0.        0.46291]
 [ 0.        0.       -0.00012]
 [ 0.        0.       -0.00001]]
```

C.1.1 Appendix: Tuning α

Tuning α to 0.005, 0.05, and 0,1.

```
################alpha=0.005,Call , K=2950####################
Spread without hedging:
[102.676751820190 381.871500525781]

Bid part path-independent:
174.254764753092
[-0.65739648 -0.97240196 -0.66982771 -0.21762963 -1.01610265
   -1.01747797
 -0.68678968 -1.01747797 -0.68678968 -0.16195706 -0.68678968
   -0.16195706
 -0.02558061]

Ask part path-independent:
-190.113775870936
[ 0.61370247  0.91549147  0.62804318  0.20101425  0.98432616
    0.98232601
  0.64374686  0.98232601  0.64374686  0.11791638  0.64374686
    0.11791638
```

```
 −0.02457605]

Bid part path−dependent :
174.257006372157
[−0.65741566 −0.97228367 −0.66977119 −0.21804102 −1.01571754
   −1.01755301
 −0.65429776 −1.01753424 −0.68896645 −0.16156764 −0.65666975
    −0.1615758
 −0.02450318]

Ask part path−dependent:
−190.105503147873
[ 0.61373265  0.91559832  0.62801357  0.20082854  0.98409567
    0.98235218
  0.61598486  0.98237647  0.6498073   0.11799835  0.61578999
      0.11797619
 −0.02463438]
################alpha=0.05,Call, K=2950#####################
Spread without hedging:
[22.6258960954639 1063.46392285111]

Bid part path−independent:
124.483104532032
[−0.65224803 −0.99082615 −0.66058262 −0.15635023 −1.00159684
   −1.00125864
 −0.6710924  −1.00125864 −0.6710924  −0.14130342 −0.6710924
    −0.14130342
 −0.00097039]

Ask part path−independent:
−235.240331688927
[ 0.61155627  0.86308502  0.6327293   0.30541788  0.99840059
    0.99823994
  0.6366406   0.99823994  0.6366406   0.13768304  0.6366406
      0.13768304
 −0.00240104]

Bid part path−dependent :
124.483105833096
[−0.65271104 −0.9906666  −0.66085366 −0.1563344  −1.00159684
   −1.00138586
 −0.6710924  −1.00141037 −0.67139576 −0.14133508 −0.6710924
    −0.14133508
 −0.00098632]

Ask part path−dependent:
−235.146553736393
[ 0.61145601  0.86307589  0.63244181  0.30538215  0.99832586
    0.99817882
  0.63349271  0.99824923  0.66746997  0.137645    0.63373844
      0.13767825
```

```
 −0.00216133]
 ################alpha=0.1,Call,  K=2950#####################
 Spread  without  hedging:
[11.4330920346238  1118.14557014556]
Bid  part  path−independent:
112.630941985136
[−0.65798133  −0.96682337  −0.67323374  −0.14600449  −0.99999838
    −0.99588668
 −0.69856026  −0.99588668  −0.69856026  −0.06345227  −0.69856026
     −0.06345227
 −0.01303577]

Ask  part  path−independent:
−257.319901669668
[ 0.61035815   0.84861051   0.62204528   0.32558544   0.9991845
      0.99731807
  0.63468436   0.99731807   0.63468436   0.13926033   0.63468436
       0.13926033
 −0.00119487]

Bid  part  path−dependent :
112.630941985136
[−0.65798133  −0.96682337  −0.67323374  −0.14600449  −0.99999838
    −0.99588668
 −0.69856026  −0.99588668  −0.69856026  −0.06345227  −0.69856026
     −0.06345227
 −0.01303577]

Ask  part  path−dependent:
−257.314151565925
[ 0.6103489    0.84860742   0.6236763    0.32557648   0.99920331
    0.99924082
  0.63461355   0.99949867   0.66833665   0.1392553    0.63461334
       0.13860218
 −0.00121844]
```

C.2 With bid-ask spread in the underlying

```
############alpha=0.001,K=2700,ttm=29  days#######
K=2700,call ,alpha=0.001  put
Spread  without  hedging:  Spread  without  hedging:
[228.029422631654  281.605020645786]      [14.7688437390495
   17.1700389340510]
Bid  part  path−independent:       Bid  part  path−independent:
240.2196517       15.34771275
[[ 0.00001  −1.27171  −1.37049]    [[0.06345  0.00766  0.00174]
 [ 0.       −0.77013  −1.40172]     [0.       0.06345  0.00572]
 [ 0.       −1.14119  −0.8392 ]     [0.       0.31032  0.0036 ]
 [ 0.        0.       −1.40172]     [0.       0.        0.00572]
```

```
 [ 0.        0.        -0.8392  ]     [0.        0.        0.0036  ]
 [ 0.        0.        -0.60746]      [0.        0.        0.29861]
 [ 0.        0.        -0.8392  ]     [0.        0.        0.0036  ]
 [ 0.        0.        -0.60746]      [0.        0.        0.29861]
 [ 0.        0.        -0.0315  ]]    [0.        0.        0.75072]]
Ask part path-independent:            Ask part path-independent:
255.9886457       16.66135273
[[0.82395  0.9366   0.93663]          [[ 0.        -0.03267   0.00015]
 [0.       0.87572  0.93659]           [ 0.        -0.0602   -0.03969]
 [0.       0.58982  0.91767]           [ 0.        -0.30284  -0.02343]
 [0.       0.       0.93659]           [ 0.         0.       -0.03969]
 [0.       0.       0.91767]           [ 0.         0.       -0.02343]
 [0.       0.       0.56346]           [ 0.         0.       -0.24978]
 [0.       0.       0.91767]           [ 0.         0.       -0.02343]
 [0.       0.       0.56346]           [ 0.         0.       -0.24978]
 [0.       0.       0.04152]]          [ 0.         0.       -0.3298
    ]]
Bid part path-dependent :             Bid part path-dependent :
240.2197162       15.34771324
[[ 0.00001  -1.27171  -1.37049]       [[0.06345  0.00766  0.00174]
 [ 0.       -0.77013  -1.40172]        [0.       0.06345  0.00572]
 [ 0.       -1.14119  -0.8392  ]       [0.       0.31032  0.0036  ]
 [ 0.        0.       -1.40172]        [0.       0.       0.00572]
 [ 0.        0.       -0.8392  ]       [0.       0.       0.0036  ]
 [ 0.        0.       -0.60746]        [0.       0.       0.29861]
 [ 0.        0.       -0.8392  ]       [0.       0.       0.0036  ]
 [ 0.        0.       -0.60746]        [0.       0.       0.29861]
 [ 0.        0.       -0.0315  ]]      [0.       0.       0.75072]]
Ask part path-dependent:              Ask part path-dependent:
255.9886442       16.66135272
[[0.82395  0.93659  0.93661]          [[ 0.        -0.03267   0.00015]
 [0.       0.87572  0.93664]           [ 0.        -0.0602   -0.03969]
 [0.       0.58982  0.9177  ]           [ 0.        -0.30284  -0.02343]
 [0.       0.       0.93657]           [ 0.         0.       -0.03969]
 [0.       0.       0.91765]           [ 0.         0.       -0.02343]
 [0.       0.       0.56346]           [ 0.         0.       -0.24978]
 [0.       0.       0.91767]           [ 0.         0.       -0.02343]
 [0.       0.       0.56346]           [ 0.         0.       -0.24978]
 [0.       0.       0.04152]]          [ 0.         0.       -0.3298
    ]]

############alpha=0.001,K=2925,ttm=29 days#######
K=2925,call ,alpha=0.001 put
Spread without hedging: Spread without hedging:
[61.9978000421878 72.8017720293526]      [51.6419352238802
   58.7371117034831]
Bid part path-independent:            Bid part path-independent:
63.08270043       54.2726772
[[ 0.       -1.12478  -0.59484]       [[0.44009  0.14211  0.014   ]
 [ 0.       -0.85254  -1.06979]        [0.       0.44009  0.01418]
 [ 0.        0.32984   0.16492]        [0.       0.84277  0.43987]
```

```
 [ 0.         0.        -1.06979]    [0.        0.        0.01418]
 [ 0.         0.         0.16492]    [0.        0.        0.43987]
 [ 0.         0.         0.34465]    [0.        0.        0.8959 ]
 [ 0.         0.         0.16492]    [0.        0.        0.43987]
 [ 0.         0.         0.34465]    [0.        0.        0.8959 ]
 [ 0.         0.         0.34473]]   [0.        0.        0.90409]]
Ask part path-independent:          Ask part path-independent:
68.15132305       57.64017892
[[0.48964  0.85745  0.90157]        [[ 0.        -0.23728   0.02882]
 [0.        0.48965  0.88892]         [ 0.        -0.71583  -0.03155]
 [0.        0.16765  0.4896 ]         [ 0.        -1.55593  -0.76623]
 [0.        0.        0.88892]         [ 0.         0.       -0.03155]
 [0.        0.        0.4896 ]         [ 0.         0.       -0.76623]
 [0.        0.        0.00092]         [ 0.         0.       -1.30574]
 [0.        0.        0.4896 ]         [ 0.         0.       -0.76623]
 [0.        0.        0.00092]         [ 0.         0.       -1.30574]
 [0.        0.        0.00111]]        [ 0.         0.
    -0.47253]]
Bid part path-dependent :           Bid part path-dependent :
63.08272075       54.27267783
[[ 0.        -1.12478  -0.59484]    [[0.44008  0.14211  0.014   ]
 [ 0.        -0.85254  -1.06979]     [0.        0.4401   0.01418]
 [ 0.         0.32984   0.16492]     [0.        0.84277  0.43987]
 [ 0.         0.       -1.06979]     [0.        0.        0.01418]
 [ 0.         0.        0.16492]     [0.        0.        0.43988]
 [ 0.         0.        0.34465]     [0.        0.        0.8959 ]
 [ 0.         0.        0.16492]     [0.        0.        0.43987]
 [ 0.         0.        0.34465]     [0.        0.        0.8959 ]
 [ 0.         0.        0.34473]]    [0.        0.        0.90409]]
Ask part path-dependent:            Ask part path-dependent:
68.15132183       57.64017766
[[0.48965  0.85745  0.90157]        [[ 0.        -0.23728   0.02882]
 [0.        0.48965  0.88892]         [ 0.        -0.71583  -0.03155]
 [0.        0.16765  0.4896 ]         [ 0.        -1.55593  -0.76623]
 [0.        0.        0.88892]         [ 0.         0.       -0.03155]
 [0.        0.        0.4896 ]         [ 0.         0.       -0.76623]
 [0.        0.        0.00092]         [ 0.         0.       -1.30574]
 [0.        0.        0.4896 ]         [ 0.         0.       -0.76623]
 [0.        0.        0.00092]         [ 0.         0.       -1.30574]
 [0.        0.        0.00111]]        [ 0.         0.
    -0.47253]]

############alpha=0.001,K=3050,ttm=29 days#######
Spread without hedging: Spread without hedging:
[10.0733093138632 11.0616216146816]     [121.374320349144
   128.901623728110]
Bid part path-independent:          Bid part path-independent:
10.09995782       124.7716516
[[ 0.00002  -0.45619  -0.81705]    [[0.86201  0.75394  0.28134]
 [ 0.        -0.079    -0.30717]     [0.        0.91195  0.75395]
 [ 0.         0.16786  -0.0948 ]     [0.        0.92951  0.91802]
```

```
 [ 0.          0.         -0.30717]   [0.         0.         0.75395]
 [ 0.          0.         -0.0948 ]   [0.         0.         0.91802]
 [ 0.          0.          0.06561]   [0.         0.         0.92951]
 [ 0.          0.         -0.0948 ]   [0.         0.         0.91802]
 [ 0.          0.          0.06561]   [0.         0.         0.92951]
 [ 0.          0.         -1.02251]]  [0.         0.         0.93503]]
Ask part path-independent:          Ask part path-independent:
10.75405462        128.727076
[[0.10452  0.34953  0.76367]         [[ 0.00002  -0.34708  -0.10063]
 [0.       0.10436  0.32248]          [ 0.        -0.64647   0.34806]
 [0.       0.00269  0.00141]          [ 0.         0.45103  -0.44472]
 [0.       0.       0.32248]          [ 0.         0.         0.34806]
 [0.       0.       0.00141]          [ 0.         0.        -0.44472]
 [0.       0.       0.00024]          [ 0.         0.        -1.23579]
 [0.       0.       0.00141]          [ 0.         0.        -0.44472]
 [0.       0.       0.00024]          [ 0.         0.        -1.23579]
 [0.       0.       0.00034]]         [ 0.         0.
    1.20879]]
Bid part path-dependent :           Bid part path-dependent :
10.09997604        124.771652
[[ 0.         -0.45618  -0.81705]    [[0.86201  0.75395  0.28134]
 [ 0.         -0.079    -0.30717]     [0.       0.91196  0.75394]
 [ 0.          0.16785  -0.0948 ]     [0.       0.92952  0.91801]
 [ 0.          0.       -0.30717]     [0.       0.       0.75396]
 [ 0.          0.       -0.09479]     [0.       0.       0.91801]
 [ 0.          0.        0.0656 ]     [0.       0.       0.92951]
 [ 0.          0.       -0.09479]     [0.       0.       0.91802]
 [ 0.          0.        0.0656 ]     [0.       0.       0.92951]
 [ 0.          0.       -1.02251]]    [0.       0.       0.93503]]
Ask part path-dependent:            Ask part path-dependent:
10.75404604        128.7270674
[[0.10447  0.34953  0.76367]         [[ 0.00001  -0.34708  -0.10063]
 [0.       0.10441  0.32251]          [ 0.        -0.64647   0.34806]
 [0.       0.00268  0.00141]          [ 0.         0.45103  -0.44472]
 [0.       0.       0.32245]          [ 0.         0.         0.34806]
 [0.       0.       0.00141]          [ 0.         0.        -0.44472]
 [0.       0.       0.00024]          [ 0.         0.        -1.23579]
 [0.       0.       0.00141]          [ 0.         0.        -0.44472]
 [0.       0.       0.00024]          [ 0.         0.        -1.23579]
 [0.       0.       0.00034]]         [ 0.         0.
    1.20879]]

 ##########################################################
#############K=2675,alpha=0.001,ttm=109  days##########
Spread without hedging: Spread without hedging:
[288.155612415026 411.255335835259]      [41.2224883447759
   52.8299490034285]
Bid part path-independent:          Bid part path-independent:
314.1520507        44.21781841
[[ 0.         -0.86046  -1.16983]    [[0.12685  0.03391  0.00525]
 [ 0.         -0.69409  -1.06983]     [0.       0.12694  0.00542]
```

```
 [ 0.         -1.17782  -0.70968]    [0.         0.4512    0.04776]
 [ 0.          0.       -1.06983]    [0.         0.        0.00542]
 [ 0.          0.       -0.70968]    [0.         0.        0.04776]
 [ 0.          0.       -0.49989]    [0.         0.        0.45119]
 [ 0.          0.       -0.70968]    [0.         0.        0.04776]
 [ 0.          0.       -0.49989]    [0.         0.        0.45119]
 [ 0.          0.        0.04333]]   [0.         0.        0.89323]]
Ask part path-independent:          Ask part path-independent:
349.1884886        49.70063116
[[0.77875  0.93111  0.93109]        [[ 0.          0.00897  -0.05266]
 [0.        0.81552  0.93112]         [ 0.         -0.10602   0.01624]
 [0.        0.51023  0.83273]         [ 0.         -0.34731  -0.13711]
 [0.        0.        0.93112]         [ 0.          0.        0.01624]
 [0.        0.        0.83273]         [ 0.          0.       -0.13711]
 [0.        0.        0.43883]         [ 0.          0.       -0.3701  ]
 [0.        0.        0.83273]         [ 0.          0.       -0.13711]
 [0.        0.        0.43883]         [ 0.          0.       -0.3701  ]
 [0.        0.        0.00199]]        [ 0.          0.
    -1.01549]]
Bid part path-dependent :           Bid part path-dependent :
314.1521023        44.22723317
[[ 0.         -0.86046  -1.16983]    [[0.12587  0.02397  0.00069]
 [ 0.         -0.69409  -1.06983]     [0.         0.12587  0.00006]
 [ 0.         -1.17782  -0.70968]     [0.         0.45976  0.02397]
 [ 0.          0.       -1.06983]     [0.         0.        0.00009]
 [ 0.          0.       -0.70968]     [0.         0.        0.06827]
 [ 0.          0.       -0.49989]     [0.         0.        0.43099]
 [ 0.          0.       -0.70968]     [0.         0.        0.06711]
 [ 0.          0.       -0.49989]     [0.         0.        0.45977]
 [ 0.          0.        0.04333]]    [0.         0.        0.89884]]
Ask part path-dependent:            Ask part path-dependent:
349.1696844        49.70062742
[[0.77844  0.93815  0.93817]        [[ 0.          0.00897  -0.05266]
 [0.        0.81469  0.93816]         [ 0.         -0.10602   0.01624]
 [0.        0.51088  0.91299]         [ 0.         -0.34731  -0.13711]
 [0.        0.        0.90393]         [ 0.          0.        0.01624]
 [0.        0.        0.8162 ]         [ 0.          0.       -0.13711]
 [0.        0.        0.43859]         [ 0.          0.       -0.3701  ]
 [0.        0.        0.81614]         [ 0.          0.       -0.13711]
 [0.        0.        0.43859]         [ 0.          0.       -0.3701  ]
 [0.        0.        0.00066]]        [ 0.          0.
    -1.01549]]

##############K=2950,alpha=0.001,ttm=109 days##########
K=2950,call,alpha=0.001 put
Spread without hedging: Spread without hedging:
[112.362199622450 153.916830172812]       [97.3839965638334
    122.966450334900]
Bid part path-independent:          Bid part path-independent:
122.7607378        106.4084836
[[ 0.0027   -0.85971  -0.89421]    [[0.43008  0.14412  0.00131]
```

```
 [ 0.         -0.46391  -0.87852]     [0.         0.43388  0.04016]
 [ 0.         -0.06122  -0.46391]     [0.         0.83101  0.43393]
 [ 0.          0.        -0.87852]     [0.         0.        0.04016]
 [ 0.          0.        -0.46391]     [0.         0.        0.43393]
 [ 0.          0.        -0.00837]     [0.         0.        0.89963]
 [ 0.          0.        -0.46391]     [0.         0.        0.43393]
 [ 0.          0.        -0.00837]     [0.         0.        0.89963]
 [ 0.          0.        -0.0243  ]]   [0.         0.        0.89963]]
Ask part path-independent:         Ask part path-independent:
134.6525825        127.9021607
[[0.50581  0.85673  0.9025  ]        [[ 0.00001  -1.14286   0.39362]
 [0.        0.50581  0.87155]          [ 0.        -0.27824  -1.07226]
 [0.        0.18006  0.48886]          [ 0.        -1.1947   -1.06177]
 [0.        0.        0.87155]          [ 0.         0.       -1.07226]
 [0.        0.        0.48886]          [ 0.         0.       -1.06177]
 [0.        0.        0.00008]          [ 0.         0.       -0.84986]
 [0.        0.        0.48886]          [ 0.         0.       -1.06177]
 [0.        0.        0.00008]          [ 0.         0.       -0.84986]
 [0.        0.        0.00109]]         [ 0.         0.
    -0.98357]]
Bid part path-dependent :          Bid part path-dependent :
122.8168005        106.4134879
[[ 0.00001  -0.85968  -0.89411]     [[0.43027  0.14519  0.00074]
 [ 0.        -0.4646   -0.87838]      [0.        0.43222  0.03943]
 [ 0.        -0.06126  -0.46394]      [0.        0.83127  0.39871]
 [ 0.         0.       -0.87838]      [0.        0.        0.03937]
 [ 0.         0.       -0.46348]      [0.        0.        0.43222]
 [ 0.         0.       -0.00824]      [0.        0.        0.90031]
 [ 0.         0.       -0.4636  ]     [0.        0.        0.50116]
 [ 0.         0.       -0.00821]      [0.        0.        0.90031]
 [ 0.         0.       -0.02419]]     [0.        0.        0.89996]]
Ask part path-dependent:           Ask part path-dependent:
134.6525815        127.9020429
[[0.50581  0.85673  0.9025  ]        [[ 0.00001  -1.14286   0.39362]
 [0.        0.50581  0.87155]          [ 0.        -0.27825  -1.07226]
 [0.        0.18006  0.48886]          [ 0.        -1.1947   -1.06177]
 [0.        0.        0.87155]          [ 0.         0.       -1.07226]
 [0.        0.        0.48886]          [ 0.         0.       -1.06177]
 [0.        0.        0.00008]          [ 0.         0.       -0.84986]
 [0.        0.        0.48886]          [ 0.         0.       -1.06177]
 [0.        0.        0.00008]          [ 0.         0.       -0.84986]
 [0.        0.        0.00109]]         [ 0.         0.
    -0.98357]]

#############K=3150,alpha=0.001,ttm=109 days##########
K=3150,call ,alpha=0.001
Spread without hedging: Spread without hedging:
[32.8511904156012  42.9572601208184]      [197.651911776914
   236.435444892533]
Bid part path-independent:         Bid part path-independent:
33.24491483        214.0055365
```

```
[[ 0.00001  -0.84989  -0.77063]    [[0.74828  0.49211  0.035    ]
 [ 0.       -0.51585  -0.48951]     [0.       0.82633  0.49186]
 [ 0.       -0.20402  -0.03823]     [0.       0.93289  0.91544]
 [ 0.        0.       -0.48951]     [0.       0.       0.49186]
 [ 0.        0.       -0.03823]     [0.       0.       0.91544]
 [ 0.        0.       -0.33336]     [0.       0.       0.93288]
 [ 0.        0.       -0.03823]     [0.       0.       0.91544]
 [ 0.        0.       -0.33336]     [0.       0.       0.93288]
 [ 0.        0.       -0.04501]]    [0.       0.       0.93283]]
Ask part path-independent:         Ask part path-independent:
39.09631099       226.4369214
[[0.19134  0.5065   0.90214]       [[ 0.        -0.69809  -0.1457  ]
 [0.       0.19131  0.50648]        [ 0.        -0.89192  -1.03037]
 [0.       0.00791  0.00788]        [ 0.        -1.26229  -0.99924]
 [0.       0.       0.50648]        [ 0.         0.       -1.03037]
 [0.       0.       0.00788]        [ 0.         0.       -0.99924]
 [0.       0.       0.00209]        [ 0.         0.       -0.6097  ]
 [0.       0.       0.00788]        [ 0.         0.       -0.99924]
 [0.       0.       0.00209]        [ 0.         0.       -0.6097  ]
 [0.       0.       0.00379]]       [ 0.         0.
     -1.35801]]
Bid part path-dependent :          Bid part path-dependent :
33.24495426       214.0055407
[[ 0.00001  -0.84988  -0.77063]    [[0.74827  0.49211  0.035    ]
 [ 0.       -0.51585  -0.48951]     [0.       0.82633  0.49186]
 [ 0.       -0.20402  -0.03823]     [0.       0.93283  0.91543]
 [ 0.        0.       -0.48951]     [0.       0.       0.49186]
 [ 0.        0.       -0.03823]     [0.       0.       0.9154  ]
 [ 0.        0.       -0.33336]     [0.       0.       0.93285]
 [ 0.        0.       -0.03823]     [0.       0.       0.91548]
 [ 0.        0.       -0.33336]     [0.       0.       0.93293]
 [ 0.        0.       -0.04501]]    [0.       0.       0.93285]]
Ask part path-dependent:           Ask part path-dependent:
39.09630488       226.4368731
[[0.19133  0.5065   0.90214]       [[ 0.        -0.69809  -0.1457  ]
 [0.       0.19131  0.5065  ]        [ 0.        -0.89192  -1.03037]
 [0.       0.00791  0.00788]        [ 0.        -1.26229  -0.99924]
 [0.       0.       0.50647]        [ 0.         0.       -1.03037]
 [0.       0.       0.00789]        [ 0.         0.       -0.99924]
 [0.       0.       0.00209]        [ 0.         0.       -0.6097  ]
 [0.       0.       0.00788]        [ 0.         0.       -0.99924]
 [0.       0.       0.00209]        [ 0.         0.       -0.6097  ]
 [0.       0.       0.00379]]       [ 0.         0.
     -1.35801]]

#############K=2675,alpha=0.001,ttm=591 days##########
K=2675,call ,alpha=0.001 put
Spread without hedging: Spread without hedging:
[420.896684303086 1144.21201744102]      [111.885300816518
   177.240357661079]
Bid part path-independent:         Bid part path-independent:
```

```
522.3864847       127.9204954
[[ 0.         -0.47166  -0.82979]    [[0.17204  0.046    0.00003]
 [ 0.         -0.5624   -0.78119]     [0.       0.17205  0.00136]
 [ 0.          0.0334   -0.66032]     [0.       0.5581   0.17195]
 [ 0.          0.       -0.78119]     [0.       0.       0.00136]
 [ 0.          0.       -0.66032]     [0.       0.       0.17195]
 [ 0.          0.       -0.08761]     [0.       0.       0.60017]
 [ 0.          0.       -0.66032]     [0.       0.       0.17195]
 [ 0.          0.       -0.08761]     [0.       0.       0.60017]
 [ 0.          0.        0.13363]]    [0.       0.       0.90008]]
Ask part path-independent:           Ask part path-independent:
681.1387663       164.8238364
[[0.76306  0.94691  0.9474 ]         [[ 0.        -0.01495   0.31476]
 [0.       0.76307  0.94699]          [ 0.        -0.30405  -0.10782]
 [0.       0.45271  0.76312]          [ 0.        -0.71443  -0.35201]
 [0.       0.       0.94699]          [ 0.         0.       -0.10782]
 [0.       0.       0.76312]          [ 0.         0.       -0.35201]
 [0.       0.       0.28243]          [ 0.         0.       -0.56908]
 [0.       0.       0.76312]          [ 0.         0.       -0.35201]
 [0.       0.       0.28243]          [ 0.         0.       -0.56908]
 [0.       0.       0.01449]]         [ 0.         0.
    -0.28987]]
Bid part path-dependent :            Bid part path-dependent :
522.3865453       127.9205069
[[ 0.         -0.47166  -0.82979]    [[0.17205  0.04601  0.00003]
 [ 0.         -0.5624   -0.78119]     [0.       0.17204  0.00136]
 [ 0.          0.0334   -0.66032]     [0.       0.5581   0.17195]
 [ 0.          0.       -0.78119]     [0.       0.       0.00136]
 [ 0.          0.       -0.66032]     [0.       0.       0.17196]
 [ 0.          0.       -0.08761]     [0.       0.       0.60017]
 [ 0.          0.       -0.66032]     [0.       0.       0.17195]
 [ 0.          0.       -0.08761]     [0.       0.       0.60017]
 [ 0.          0.        0.13363]]    [0.       0.       0.90008]]
Ask part path-dependent:             Ask part path-dependent:
681.1383415       164.8234917
[[0.76311  0.94695  0.94736]         [[ 0.        -0.01495   0.31476]
 [0.       0.76311  0.94695]          [ 0.        -0.30405  -0.10782]
 [0.       0.45271  0.76315]          [ 0.        -0.71442  -0.35201]
 [0.       0.       0.94695]          [ 0.         0.       -0.10782]
 [0.       0.       0.76303]          [ 0.         0.       -0.35201]
 [0.       0.       0.28243]          [ 0.         0.       -0.56909]
 [0.       0.       0.76308]          [ 0.         0.       -0.35201]
 [0.       0.       0.28243]          [ 0.         0.       -0.56909]
 [0.       0.       0.01448]]         [ 0.         0.
    -0.28987]]
#############K=3000,alpha=0.001,ttm=591 days##########
K=3000,call ,alpha=0.001  put
Spread without hedging: Spread without hedging:
[245.158027732903 554.205318102853]      [159.921172649136
   271.245376211055]
Bid part path-independent:           Bid part path-independent:
```

```
299.4633392        191.3675494
[[ 0.         -0.6814    -0.79563]    [[0.32399  0.08871  0.00002]
 [ 0.         -0.62035  -0.79758]     [0.       0.3374   0.01424]
 [ 0.          0.21852  -0.61945]     [0.       0.78279  0.33793]
 [ 0.          0.       -0.79758]     [0.       0.       0.01424]
 [ 0.          0.       -0.61945]     [0.       0.       0.33793]
 [ 0.          0.        0.20256]     [0.       0.       0.89967]
 [ 0.          0.       -0.61945]     [0.       0.       0.33793]
 [ 0.          0.        0.20256]     [0.       0.       0.89967]
 [ 0.          0.       -0.16999]]    [0.       0.       0.90025]]
Ask part path-independent:         Ask part path-independent:
378.7427554        234.4107883
[[0.62173  0.91218  0.91594]       [[ 0.         -0.01092   0.10531]
 [0.       0.62173  0.91218]        [ 0.         -0.33261  -0.03625]
 [0.       0.25027  0.57015]        [ 0.         -0.76754  -0.35819]
 [0.       0.       0.91218]        [ 0.          0.       -0.03625]
 [0.       0.       0.57015]        [ 0.          0.       -0.35819]
 [0.       0.       0.00059]        [ 0.          0.       -0.91571]
 [0.       0.       0.57015]        [ 0.          0.       -0.35819]
 [0.       0.       0.00059]        [ 0.          0.       -0.91571]
 [0.       0.       0.00084]]       [ 0.          0.
    -0.83591]]
Bid part path-dependent :          Bid part path-dependent :
299.4636637        191.4007598
[[ 0.         -0.6814    -0.79563]    [[0.32417  0.08918  0.00199]
 [ 0.         -0.62035  -0.79758]     [0.       0.33407  0.0162 ]
 [ 0.          0.21852  -0.61945]     [0.       0.78088  0.30551]
 [ 0.          0.       -0.79758]     [0.       0.       0.01635]
 [ 0.          0.       -0.61945]     [0.       0.       0.33408]
 [ 0.          0.        0.20256]     [0.       0.       0.8992 ]
 [ 0.          0.       -0.61945]     [0.       0.       0.41848]
 [ 0.          0.        0.20256]     [0.       0.       0.8992 ]
 [ 0.          0.       -0.16998]]    [0.       0.       0.8988 ]]
Ask part path-dependent:           Ask part path-dependent:
378.7427517        234.4107674
[[0.62173  0.91218  0.91594]       [[ 0.         -0.01092   0.10531]
 [0.       0.62173  0.91218]        [ 0.         -0.33261  -0.03625]
 [0.       0.25027  0.57015]        [ 0.         -0.76754  -0.35819]
 [0.       0.       0.91218]        [ 0.          0.       -0.03625]
 [0.       0.       0.57015]        [ 0.          0.       -0.35819]
 [0.       0.       0.00059]        [ 0.          0.       -0.91571]
 [0.       0.       0.57015]        [ 0.          0.       -0.35819]
 [0.       0.       0.00059]        [ 0.          0.       -0.91571]
 [0.       0.       0.00084]]       [ 0.          0.
    -0.83591]]

#############K=3400,alpha=0.001,ttm=591 days##########
K=3400,call ,alpha=0.001 put
Spread without hedging: Spread without hedging:
[124.705307712226 239.782192287375]       [304.112038381576
   469.095607040746]
```

```
Bid part path-independent:
139.2320115        365.0705047
[[ 0.00003  -0.59733  -0.54942]
 [ 0.       -0.51327  -0.71903]
 [ 0.       -0.4647    0.00336]
 [ 0.        0.       -0.71903]
 [ 0.        0.        0.00336]
 [ 0.        0.       -0.19228]
 [ 0.        0.        0.00336]
 [ 0.        0.       -0.19228]
 [ 0.        0.        0.06969]]
Ask part path-independent:
179.0464394        423.1269721
[[0.36161  0.66389  0.91186]
 [0.       0.36013  0.66068]
 [0.       0.08977  0.13535]
 [0.       0.       0.66068]
 [0.       0.       0.13535]
 [0.       0.       0.00015]
 [0.       0.       0.13535]
 [0.       0.       0.00015]
 [0.       0.       0.00343]]
    -0.59008]]
Bid part path-dependent :
139.2346858        365.1023667
[[ 0.       -0.59734  -0.54944]
 [ 0.       -0.51324  -0.71901]
 [ 0.       -0.46467   0.00335]
 [ 0.        0.       -0.71901]
 [ 0.        0.        0.00334]
 [ 0.        0.       -0.19226]
 [ 0.        0.        0.00334]
 [ 0.        0.       -0.19226]
 [ 0.        0.        0.06968]]
Ask part path-dependent:
178.980023         423.126693
[[0.36155  0.67673  0.91154]
 [0.       0.36023  0.67673]
 [0.       0.08005  0.152   ]
 [0.       0.       0.61471]
 [0.       0.       0.15225]
 [0.       0.       0.00072]
 [0.       0.       0.08005]
 [0.       0.       0.00058]
 [0.       0.       0.00054]]
    -0.59009]]
```

```
Bid part path-independent:
[[0.58061  0.31444  0.01252]
 [0.       0.67524  0.3068  ]
 [0.       0.90729  0.78517]
 [0.       0.       0.3068  ]
 [0.       0.       0.78517]
 [0.       0.       0.90729]
 [0.       0.       0.78517]
 [0.       0.       0.90729]
 [0.       0.       0.90936]]
Ask part path-independent:
[[ 0.00001  -0.15549  -0.14496]
 [ 0.       -0.47644  -0.1029  ]
 [ 0.       -0.59428  -0.44611]
 [ 0.        0.       -0.1029  ]
 [ 0.        0.       -0.44611]
 [ 0.        0.       -0.59535]
 [ 0.        0.       -0.44611]
 [ 0.        0.       -0.59535]
 [ 0.        0.
Bid part path-dependent :
[[0.58209  0.31168  0.00122]
 [0.       0.67489  0.31147]
 [0.       0.91282  0.77326]
 [0.       0.       0.30948]
 [0.       0.       0.77378]
 [0.       0.       0.89984]
 [0.       0.       0.87497]
 [0.       0.       0.91282]
 [0.       0.       0.91276]]
Ask part path-dependent:
[[ 0.00001  -0.15549  -0.14496]
 [ 0.       -0.47644  -0.1029  ]
 [ 0.       -0.59428  -0.44611]
 [ 0.        0.       -0.1029  ]
 [ 0.        0.       -0.44611]
 [ 0.        0.       -0.59535]
 [ 0.        0.       -0.44611]
 [ 0.        0.       -0.59535]
 [ 0.        0.
```

C.2.1 Further tuning α

We tune the α to 0.01 to see if there's more prominent discrepancy.

```
K=3400,call ,alpha=0.01   put
Spread  without  hedging: Spread  without  hedging:
[41.1012972371902  1146.68837014656]      [80.0531899779322
   1085.40405797048]
Bid  part  path-independent:        Bid  part  path-independent:
70.23979955       334.250769
[[ 0.00003  -0.94369  -0.99357]    [[0.65603  0.24199  0.00045]
 [ 0.      -0.21867  -0.65764]     [0.       0.75002  0.21615]
 [ 0.      -0.18863  -0.17339]     [0.       0.95788  0.86045]
 [ 0.       0.       -0.65764]     [0.       0.       0.21615]
 [ 0.       0.       -0.17339]     [0.       0.       0.86045]
 [ 0.       0.        0.15526]     [0.       0.       0.99008]
 [ 0.       0.       -0.17339]     [0.       0.       0.86045]
 [ 0.       0.        0.15526]     [0.       0.       0.99008]
 [ 0.       0.       -0.2211  ]]   [0.       0.       0.98981]]
Ask  part  path-independent:        Ask  part  path-independent:
208.2734201       614.5963534
[[0.42956  0.69965  1.00277]       [[ 0.       -0.58552   0.07728]
 [0.       0.36863  0.68131]        [ 0.       -0.8127   -0.43319]
 [0.       0.0635   0.14835]        [ 0.       -0.92291  -0.83177]
 [0.       0.       0.68131]        [ 0.        0.       -0.43319]
 [0.       0.       0.14835]        [ 0.        0.       -0.83177]
 [0.       0.       0.00012]        [ 0.        0.       -0.98079]
 [0.       0.       0.14835]        [ 0.        0.       -0.83177]
 [0.       0.       0.00012]        [ 0.        0.       -0.98079]
 [0.       0.       0.00009]]       [ 0.        0.
    -0.88081]]
Bid  part  path-dependent  :        Bid  part  path-dependent  :
70.24889797       334.2528231
[[ 0.      -0.94366  -0.99357]     [[0.65588  0.24199  0.00021]
 [ 0.      -0.21869  -0.65765]     [0.       0.75005  0.21613]
 [ 0.      -0.18861  -0.17339]     [0.       0.95773  0.86011]
 [ 0.       0.       -0.65765]     [0.       0.       0.21613]
 [ 0.       0.       -0.17337]     [0.       0.       0.85973]
 [ 0.       0.        0.15523]     [0.       0.       0.98994]
 [ 0.       0.       -0.17337]     [0.       0.       0.87451]
 [ 0.       0.        0.15523]     [0.       0.       0.98993]
 [ 0.       0.       -0.22108]]    [0.       0.       0.98993]]
Ask  part  path-dependent:          Ask  part  path-dependent:
208.2675762       614.5952339
[[0.42959  0.69988  1.00232]       [[ 0.       -0.58552   0.07728]
 [0.       0.36861  0.68401]        [ 0.       -0.8127   -0.43319]
 [0.       0.06315  0.15146]        [ 0.       -0.92291  -0.83177]
 [0.       0.       0.67881]        [ 0.        0.       -0.43319]
 [0.       0.       0.15135]        [ 0.        0.       -0.83177]
 [0.       0.       0.00008]        [ 0.        0.       -0.98079]
 [0.       0.       0.14275]        [ 0.        0.       -0.83177]
 [0.       0.       0.00008]        [ 0.        0.       -0.98079]
 [0.       0.       0.00046]]       [ 0.        0.
    -0.88081]]
```

```
K=3400,call ,alpha=0.01   put
Spread  without  hedging: Spread  without  hedging:
[128.583056867595 137.683489121961]      [396.514681513536
   414.611859223663]
Bid  part  path−independent:       Bid  part  path−independent:
132.5901558       405.0719549
[[−0.35009  −0.66464  −0.99997]   [[0.65682  0.3415   0.00121]
 [  0.        −0.25082  −0.64739]   [0.        0.75857  0.36132]
 [  0.        −0.02324  −0.0688  ]   [0.        0.9817   0.94539]
 [  0.         0.        −0.64739]   [0.        0.        0.36132]
 [  0.         0.        −0.0688  ]   [0.        0.        0.94539]
 [  0.         0.         0.00011]   [0.        0.        1.00043]
 [  0.         0.        −0.0688  ]   [0.        0.        0.94539]
 [  0.         0.         0.00011]   [0.        0.        1.00043]
 [  0.         0.        −0.       ]]  [0.        0.        1.       ]]
Ask  part  path−independent:       Ask  part  path−independent:
133.4295912       405.8928119
[[0.35109  0.66393  1.00152]       [[−0.65566  −0.34176  −0.00037]
 [0.        0.25241  0.64577]        [  0.        −0.75692  −0.36312]
 [0.        0.02282  0.06878]        [  0.        −0.98166  −0.94576]
 [0.        0.        0.64577]        [  0.         0.        −0.36312]
 [0.        0.        0.06878]        [  0.         0.        −0.94576]
 [0.        0.        0.00024]        [  0.         0.        −0.99924]
 [0.        0.        0.06878]        [  0.         0.        −0.94576]
 [0.        0.        0.00024]        [  0.         0.        −0.99924]
 [0.        0.        0.       ]]       [  0.         0.        −1.
    ]]
 #alpha=0.0001##################
Call  Put
Bid  part  path−dependent  :       Bid  part  path−dependent  :
132.5901558       405.0719557
[[−0.35009  −0.66464  −0.99997]   [[0.65646  0.34138  0.00074]
 [  0.        −0.25082  −0.64739]   [0.        0.75845  0.36132]
 [  0.        −0.02324  −0.0688  ]   [0.        0.98158  0.94539]
 [  0.         0.        −0.64739]   [0.        0.        0.36132]
 [  0.         0.        −0.0688  ]   [0.        0.        0.94539]
 [  0.         0.         0.00011]   [0.        0.        1.00031]
 [  0.         0.        −0.0688  ]   [0.        0.        0.94539]
 [  0.         0.         0.00011]   [0.        0.        1.00031]
 [  0.         0.        −0.       ]]  [0.        0.        1.       ]]
Ask  part  path−dependent:         Ask  part  path−dependent:
133.4295909       405.8928119
[[0.35119  0.66393  1.00152]       [[−0.65566  −0.34176  −0.00037]
 [0.        0.25241  0.64587]        [  0.        −0.75692  −0.36312]
 [0.        0.02293  0.06878]        [  0.        −0.98166  −0.94576]
 [0.        0.        0.64587]        [  0.         0.        −0.36312]
 [0.        0.        0.06878]        [  0.         0.        −0.94576]
 [0.        0.        0.00024]        [  0.         0.        −0.99924]
 [0.        0.        0.06878]        [  0.         0.        −0.94576]
 [0.        0.        0.00024]        [  0.         0.        −0.99924]
```

```
[0.          0.          0.        ]]          [  0.          0.          −1.
    ]]
```

Bibliography

Black et al. (1973). "The pricing of options and corporate liabilities". In: *Journal of Political Economy* 81.3, pp. 637–654.

Boyle, Phelim P (1986). "Option valuation using a tree-jump process". In: *International Options Journal* 3, pp. 7–12.

Carr et al. (2001). "Pricing and hedging in incomplete markets". In: *Journal of financial Economics* 62.1, pp. 131–167.

Chateauneuf et al. (1996). "Choquet Pricing For Financial Markets With Frictions 1". In: *Mathematical Finance* 6.3, pp. 323–330.

Choi et al. (1988). "On the estimation of bid-ask spreads: Theory and evidence". In: *Journal of Financial and Quantitative Analysis* 23.2, pp. 219–230.

Cox, John C et al. (1979). "Option pricing: A simplified approach". In: *Journal of financial Economics* 7.3, pp. 229–263.

Dalang et al. (1990). "Equivalent martingale measures and no-arbitrage in stochastic securities market models". In: *Stochastics: An International Journal of Probability and Stochastic Processes* 29.2, pp. 185–201.

Delbaen, Freddy and Walter Schachermayer (1994). "A general version of the fundamental theorem of asset pricing". In: *Mathematische Annalen* 300.1, pp. 463–520.

Figlewski, Stephen and Bin Gao (1999). "The adaptive mesh model: a new approach to efficient option pricing". In: *Journal of Financial Economics* 53.3, pp. 313–351.

Follmer, H and A Schied. *Stochastic Finance: An Introduction In Discrete Time 2, 2004*.

Föllmer, Hans and Alexander Schied (2011). *Stochastic finance: an introduction in discrete time*. Walter de Gruyter.

George et al. (1991). "Estimation of the bid–ask spread and its components: A new approach". In: *The Review of Financial Studies* 4.4, pp. 623–656.

Gould, John P and Dan Galai (1974). "Transactions costs and the relationship between put and call prices". In: *Journal of Financial Economics* 1.2, pp. 105–129.

Harrison, J Michael and David M Kreps (1979). "Martingales and arbitrage in multiperiod securities markets". In: *Journal of Economic theory* 20.3, pp. 381–408.

Huang, Roger D and Hans R Stoll (1997). "The components of the bid-ask spread: A general approach". In: *The Review of Financial Studies* 10.4, pp. 995–1034.

Hull, John and Alan White (1988). "The use of the control variate technique in option pricing". In: *Journal of Financial and Quantitative analysis* 23.3, pp. 237–251.

Hull, John C (2003). *Options futures and other derivatives*. Pearson Education India.

Jackwerth, Jens Carsten (1999). "Option-implied risk-neutral distributions and implied binomial trees: A literature review". In: *The Journal of Derivatives* 7.2, pp. 66–82.

Jarrow, Robert A and Andrew T Rudd (1983). *Option pricing*.

Kahneman, Daniel and Amos Tversky (1979). "Prospect theory: An analysis of decision under risk". In: *Econometrica* 47.2, pp. 363–391.

Kamara, Avraham and Thomas W Miller (1995). "Daily and intradaily tests of European put-call parity". In: *Journal of Financial and Quantitative Analysis* 30.4, pp. 519–539.

Klemkosky, Robert C and Bruce G Resnick (1980). "An ex ante analysis of put-call parity". In: *Journal of Financial Economics* 8.4, pp. 363–378.

Kraft, Dieter (1988). "A software package for sequential quadratic programming". In: *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*.

Madan et al. (1998). "The variance gamma process and option pricing". In: *Review of Finance* 2.1, pp. 79–105.

Madan, Dilip and Wim Schoutens (2016). *Applied Conic Finance*. Cambridge University Press.

Madan, Dilip B and Alexander Cherny (2010). "Markets as a counterparty: an introduction to conic finance". In: *International Journal of Theoretical and Applied Finance* 13.08, pp. 1149–1177.

Merton, Robert C (1973). "Theory of rational option pricing". In: *Theory of Valuation*, pp. 229–288.

R Grimwood, S Hodges (2000). "Taxonomy of algorithms". In:

Rendleman Jr, Richard J and Brit J Bartter (1979). "Two-state option pricing". In: *The Journal of Finance* 34.5, pp. 1093–1110.

Roll, Richard (1984). "A simple implicit measure of the effective bid-ask spread in an efficient market". In: *The Journal of Finance* 39.4, pp. 1127–1139.

Ross, Stephen A et al. (1973). *Return, risk and arbitrage*. Rodney L. White Center for Financial Research, The Wharton School ...

Schmeidler, David (1989). "Subjective probability and expected utility without additivity". In: *Econometrica: Journal of the Econometric Society*, pp. 571–587.

Stoll, Hans R (1969). "The relationship between put and call option prices". In: *The Journal of Finance* 24.5, pp. 801–824.

— (1989). "Inferring the components of the bid-ask spread: Theory and empirical tests". In: *the Journal of Finance* 44.1, pp. 115–134.

Trigeorgis, Lenos (1991). "A log-transformed binomial numerical analysis method for valuing complex multi-option investments". In: *Journal of Financial and Quantitative Analysis* 26.3, pp. 309–326.

Tversky, Amos and Daniel Kahneman (1992). "Advances in prospect theory: Cumulative representation of Uncertainty". In: *Journal of Risk and Uncertainty* 5.4, pp. 297–323.

Verschuren, Piet, Hans Doorewaard, and Michelle Mellion (2010). *Designing a research project*. Vol. 2. Eleven International Publishing The Hague.

Webster, Jane and Richard T. Watson (2002). "Analyzing the Past to Prepare for the Future: Writing a Literature Review". In: *MIS Quarterly* 26.2.